Admin Mod Kompendium

Falko Hartmann (aka [WING] Black Knight)

10.03.2012

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Vorwort . . . 1
1.2 Was ist Admin Mod? . . . 2
1.3 Warum nicht ein anderes Plugin nutzen? . . . 3

2 Historie **5**

3 Installation **7**
3.1 Voraussetzungen . . . 7
3.2 Installation (Windows) . . . 7
3.3 Installation (Linux/FreeBSD) . . . 9
3.4 MySQL/PostgreSQL Installation . . . 11
3.5 Verzeichnisstruktur . . . 13
3.6 Admin Mod mit Bots . . . 13

4 Konfiguration **15**
4.1 Admin Mod einrichten (adminmod.cfg) . . . 15
4.1.1 admin_balance_teams . . . 15
4.1.2 admin_bot_protection . . . 16
4.1.3 admin_connect_msg . . . 16
4.1.4 admin_cs_restrict . . . 17
4.1.5 admin_debug . . . 17
4.1.6 admin_devel . . . 18
4.1.7 admin_fun_mode . . . 18
4.1.8 admin_fx . . . 19
4.1.9 admin_gag_name . . . 19
4.1.10 admin_gag_sayteam . . . 20
4.1.11 admin_highlander . . . 20
4.1.12 admin_ignore_immunity . . . 21
4.1.13 admin_plugin_file . . . 21
4.1.14 admin_reconnect_timeout . . . 22
4.1.15 admin_reject_msg . . . 22
4.1.16 admin_repeat_freq . . . 23
4.1.17 admin_repeat_msg . . . 23
4.1.18 admin_quiet . . . 24

4.1.19 admin_vault_file 24
4.1.20 admin_mod_version 25
4.1.21 admin_vote_autostart 25
4.1.22 admin_vote_echo 26
4.1.23 admin_vote_freq 26
4.1.24 admin_vote_maxextend 27
4.1.25 admin_vote_ratio 27
4.1.26 allow_client_exec 28
4.1.27 ami_sv_maxplayers 28
4.1.28 amv_anti_cheat_options 29
4.1.29 amv_default_config_dir 29
4.1.30 amv_enable_beta 30
4.1.31 amv_hide_reserved_slots 31
4.1.32 amv_log_passwords 31
4.1.33 amv_private_server 32
4.1.34 amv_prvt_kick_message 32
4.1.35 amv_reconnect_time 33
4.1.36 amv_register_cmds 33
4.1.37 amv_vote_duration 34
4.1.38 default_access 34
4.1.39 encrypt_password 35
4.1.40 file_access_read 36
4.1.41 file_access_write 36
4.1.42 help_file (veraltet) 37
4.1.43 ips_file 37
4.1.44 kick_ratio 38
4.1.45 map_ratio 38
4.1.46 maps_file 39
4.1.47 models_file 39
4.1.48 models_kick_msg 40
4.1.49 mysql_database (nur MySQL-Version) 40
4.1.50 mysql_dbtable_ips (nur MySQL-Version) 41
4.1.51 mysql_dbtable_models (nur MySQL-Version) 41
4.1.52 mysql_dbtable_plugins (nur MySQL-Version) 42
4.1.53 mysql_dbtable_tags (nur MySQL-Version) 42
4.1.54 mysql_dbtable_users (nur MySQL-Version) 43
4.1.55 mysql_dbtable_words (nur MySQL-Version) 43
4.1.56 mysql_host (nur MySQL-Version) 44
4.1.57 mysql_pass (nur MySQL-Version) 44
4.1.58 mysql_preload (nur MySQL-Version) 45
4.1.59 mysql_tags_sql (nur MySQL-Version) 45
4.1.60 mysql_user (nur MySQL-Version) 46
4.1.61 mysql_users_sql (nur MySQL-Version) 46
4.1.62 nicks_kick_msg 47

4.1.63 password_field . . . 47
4.1.64 pgsql_database (nur PostgreSQL-Version) . . . 48
4.1.65 pgsql_dbtable_ips (nur PostgreSQL-Version) . . . 48
4.1.66 pgsql_dbtable_models (nur PostgreSQL-Version) . . . 49
4.1.67 pgsql_dbtable_plugins (nur PostgreSQL-Version) . . . 49
4.1.68 pgsql_dbtable_tags (nur PostgreSQL-Version) . . . 50
4.1.69 pgsql_dbtable_users (nur PostgreSQL-Version) . . . 50
4.1.70 pgsql_dbtable_words (nur PostgreSQL-Version) . . . 51
4.1.71 pgsql_host (nur PostgreSQL-Version) . . . 51
4.1.72 pgsql_pass (nur PostgreSQL-Version) . . . 52
4.1.73 pgsql_port (nur PostgreSQL-Version) . . . 52
4.1.74 pgsql_preload (nur PostgreSQL-Version) . . . 53
4.1.75 pgsql_tags_sql (nur PostgreSQL-Version) . . . 53
4.1.76 pgsql_user (nur PostgreSQL-Version) . . . 54
4.1.77 pgsql_users_sql (nur PostgreSQL-Version) . . . 54
4.1.78 pretty_say . . . 55
4.1.79 public_slots_free . . . 55
4.1.80 reserve_slots . . . 56
4.1.81 reserve_slots_msg . . . 56
4.1.82 reserve_type . . . 57
4.1.83 script_file (veraltet) . . . 57
4.1.84 use_regex . . . 58
4.1.85 users_file . . . 58
4.1.86 vote_freq . . . 59
4.1.87 words_file . . . 59
4.2 Plugins installieren (plugin.ini) . . . 60
4.3 Plugins kompilieren . . . 61
4.4 Administratoren einrichten (users.ini) . . . 63
4.4.1 Serverseitige Einstellungen . . . 63
4.4.1.1 *Player*:Password:Rights . . . 63
4.4.1.2 Player:*Password*:Rights . . . 64
4.4.1.3 Player:Password:*Rights* . . . 65
4.4.1.4 Serverseitige Einstellungen (adminmod.cfg) . . . 65
4.4.2 Clientseitige Einstellungen . . . 66
4.4.3 Rechtelevel . . . 68
4.4.4 Serverplätze reservieren . . . 71
4.4.4.1 reserve_type 0 . . . 71
4.4.4.2 reserve_type 1 . . . 72
4.4.4.3 reserve_type 2 . . . 72
4.4.5 Namen reservieren . . . 74
4.4.6 RegEx (Clantags reservieren etc.) . . . 74
4.4.6.1 Einführung . . . 74
4.4.6.2 Clantag reservieren . . . 75
4.5 Benutzbare Maps einstellen (maps.ini) . . . 76

4.6 IPs für reservierte Slots festlegen (ips.ini) 77
4.7 Models reservieren (models.ini) 78
4.8 Chat zensieren (wordlist.txt) 79
4.9 Pluginspezifische Einstellungen (vault.ini) 80
4.10 MySQL Installation einrichten 80
4.10.1 Datenbankeinrichtung 81
4.10.2 Users-Tabelle 81
4.10.3 Tags-Tabelle 81
4.10.4 Models-Tabelle 81
4.10.5 Ips-Tabelle 82
4.10.6 Words-Tabelle 82
4.10.7 Plugins-Tabelle 82
4.10.8 Zugriff auf die Datenbank 82
4.10.9 Außergewöhnliche Einstellungen 83
4.11 PostgreSQL Installation einrichten 83
4.11.1 Datenbankeinrichtung 84
4.11.2 Users-Tabelle 84
4.11.3 Tags-Tabelle 84
4.11.4 Models-Tabelle 85
4.11.5 Ips-Tabelle 85
4.11.6 Words-Tabelle 85
4.11.7 Plugins-Tabelle 85
4.11.8 Zugriff auf die Datenbank 86
4.11.9 Außergewöhnliche Einstellungen 86
4.12 Beispiel: Admin Mod in phpBB integrieren (MySQL) 86
4.12.1 Admin Mod einrichten 87
4.12.2 Nutzung von Views 88
4.12.3 Nutzung von phpBB2 89

5 Standardplugins und Befehle **90**
5.1 Befehle unabhängig von Plugins 90
5.1.1 admin_adm 90
5.1.2 admin_cmd 91
5.1.3 admin_command 91
5.1.4 admin_help 92
5.1.5 admin_login 92
5.1.6 admin_password 93
5.1.7 admin_status 93
5.1.8 admin_version 94
5.2 plugin_antiflood 95
5.3 plugin_base 95
5.3.1 admin_abort_vote 95
5.3.2 admin_ban 96
5.3.3 admin_banip 97

5.3.4 admin_cfg . . . 98
5.3.5 admin_chat . . . 98
5.3.6 admin_csay . . . 99
5.3.7 admin_dmesg . . . 99
5.3.8 admin_fraglimit . . . 100
5.3.9 admin_friendlyfire . . . 100
5.3.10 admin_gravity . . . 101
5.3.11 admin_hostname . . . 101
5.3.12 admin_kick . . . 102
5.3.13 admin_listmaps . . . 103
5.3.14 admin_map . . . 103
5.3.15 admin_motd . . . 104
5.3.16 admin_nextmap . . . 104
5.3.17 admin_nopass . . . 105
5.3.18 admin_pass . . . 105
5.3.19 admin_pause . . . 106
5.3.20 admin_psay . . . 106
5.3.21 admin_rcon . . . 107
5.3.22 admin_reload . . . 108
5.3.23 admin_say . . . 108
5.3.24 admin_servercfg . . . 109
5.3.25 admin_ssay . . . 109
5.3.26 admin_teamplay . . . 110
5.3.27 admin_timeleft . . . 110
5.3.28 admin_timelimit . . . 111
5.3.29 admin_tsay . . . 111
5.3.30 admin_unban . . . 112
5.3.31 admin_unpause . . . 112
5.3.32 admin_userlist . . . 113
5.3.33 admin_vote_kick . . . 113
5.3.34 admin_vote_map . . . 114
5.3.35 admin_vsay . . . 115
5.4 plugin_chat . . . 115
5.4.1 admin_messagemode . . . 116
5.4.2 admin_nomessagemode . . . 117
5.4.3 say currentmap . . . 117
5.4.4 say nextmap . . . 118
5.4.5 say timeleft . . . 118
5.5 plugin_cheat . . . 119
5.5.1 admin_godmode . . . 119
5.5.2 admin_noclip . . . 120
5.5.3 admin_stack . . . 120
5.5.4 admin_teleport . . . 121
5.5.5 admin_userorigin . . . 121

5.6 plugin_CS . . . 122
5.6.1 admin_autokick . . . 122
5.6.2 admin_autoteambalance . . . 123
5.6.3 admin_buytimer . . . 123
5.6.4 admin_c4timer . . . 124
5.6.5 admin_chattime . . . 124
5.6.6 admin_consistency . . . 125
5.6.7 admin_ct . . . 125
5.6.8 admin_fadetoblack . . . 126
5.6.9 admin_flashlight . . . 126
5.6.10 admin_footsteps . . . 127
5.6.11 admin_forcecamera . . . 127
5.6.12 admin_forcechasecam . . . 128
5.6.13 admin_freezetime . . . 128
5.6.14 admin_ghostfrequency . . . 129
5.6.15 admin_hpenalty . . . 129
5.6.16 admin_kickpercent . . . 130
5.6.17 admin_limitteams . . . 130
5.6.18 admin_maxrounds . . . 131
5.6.19 admin_mapvoteratio . . . 131
5.6.20 admin_playerid . . . 132
5.6.21 admin_restart . . . 132
5.6.22 admin_restartround . . . 133
5.6.23 admin_restrict . . . 133
5.6.24 admin_restrictmenu . . . 136
5.6.25 admin_roundtime . . . 136
5.6.26 admin_startmoney . . . 137
5.6.27 admin_t . . . 137
5.6.28 admin_tkpunish . . . 138
5.6.29 admin_unrestrict . . . 138
5.6.30 admin_vote_restart . . . 139
5.6.31 admin_winlimit . . . 140
5.7 plugin_fun . . . 140
5.7.1 admin_disco . . . 141
5.7.2 admin_fun . . . 141
5.7.3 admin_glow . . . 142
5.7.4 say glow . . . 142
5.8 plugin_hldsld_mapvote . . . 143
5.8.1 admin_cancelvote . . . 143
5.8.2 admin_denymap . . . 144
5.8.3 admin_startvote . . . 144
5.8.4 say cancelvote . . . 145
5.8.5 say denymap . . . 145
5.8.6 say mapvote . . . 146

5.8.7 say rockthevote . . . 146
5.8.8 say vote . . . 147
5.9 plugin_message . . . 147
5.10 plugin_retribution . . . 147
5.10.1 admin_bury . . . 148
5.10.2 admin_execall . . . 148
5.10.3 admin_execclient . . . 149
5.10.4 admin_execteam . . . 149
5.10.5 admin_gag . . . 150
5.10.6 admin_llama . . . 151
5.10.7 admin_slap . . . 151
5.10.8 admin_slay . . . 152
5.10.9 admin_slayteam . . . 153
5.10.10 admin_unbury . . . 153
5.10.11 admin_ungag . . . 154
5.10.12 admin_unllama . . . 154
5.11 plugin_spawn . . . 155
5.12 plugin_TFC . . . 155
5.12.1 admin_balance . . . 155
5.12.2 admin_blue . . . 156
5.12.3 admin_green . . . 156
5.12.4 admin_prematch . . . 157
5.12.5 admin_red . . . 157
5.12.6 admin_yellow . . . 158

6 FAQ **159**
6.1 admin_restartround, admin_ct oder admin_t funktionieren nicht! . . . 159
6.2 Admin Mod funktioniert nicht! . . . 159
6.3 Wo bekomme ich weitere Plugins her? . . . 160
6.4 Admin Mod Befehle funktionieren nicht (z.B. in HLSW) . . . 160
6.5 Couldn't find Meta_Query() . . . 160
6.6 WARNING: meta-interface version mismatch . . . 161

7 Erweiterungen **162**
7.1 LogDaemon (LogD) . . . 162
7.1.1 Installation . . . 162
7.1.2 Variablen . . . 163
7.1.2.1 logd_debug . . . 163
7.1.2.2 logd_version . . . 163
7.1.3 Befehle . . . 163
7.1.3.1 logd_block . . . 163
7.1.3.2 logd_reg . . . 164
7.1.3.3 logd_reginfo . . . 164
7.1.3.4 logd_status . . . 165

7.1.4 Performance und Plattenplatz . . . 166
7.2 StatsMe . . . 167
7.2.1 Installation und Konfiguration . . . 167
7.2.2 Performance und Plattenplatz . . . 167
7.3 HLStats . . . 168
7.4 Metamod Plugins . . . 168

8 Scripting **169**
8.1 Syntax . . . 169
8.2 Datentypen, Variablen und Konstanten . . . 170
8.3 Operatoren . . . 171
8.4 Verzweigungen . . . 172
8.5 Schleifen . . . 173
8.6 Direktiven . . . 174
8.6.1 include . . . 174
8.6.2 define . . . 175
8.6.3 if, elseif, else und endif . . . 175
8.7 Funktionen und Events . . . 175
8.7.1 Beispiele für Events von Admin Mod . . . 176
8.8 Tutorial . . . 176
8.8.1 Basis . . . 176
8.8.2 Befehle registrieren . . . 177
8.8.3 Auf say reagieren . . . 179
8.8.4 Spielerconnect abfangen . . . 181
8.8.5 Timer verwenden . . . 182
8.8.6 Vote ausführen . . . 183
8.8.7 Vault-File nutzen . . . 185
8.8.8 LogD verwenden . . . 186
8.8.9 Menüs nutzen . . . 189
8.9 Includes . . . 193
8.9.1 admin.inc . . . 194
8.9.2 adminlib.inc . . . 195
8.9.3 console.inc . . . 195
8.9.4 core.inc . . . 195
8.9.5 fixed.inc . . . 196
8.9.6 math.inc . . . 196
8.9.7 plugin.inc . . . 196
8.9.8 string.inc . . . 197
8.10 Funktionsreferenz . . . 198
8.10.1 access . . . 198
8.10.2 auth . . . 199
8.10.3 ban . . . 200
8.10.4 censor_words . . . 201
8.10.5 centersay . . . 202

8.10.6 centersayex . . . 203
8.10.7 changelevel . . . 204
8.10.8 ChangeMap . . . 204
8.10.9 check_auth . . . 205
8.10.10 check_immunity . . . 206
8.10.11 check_param . . . 207
8.10.12 check_user . . . 208
8.10.13 check_words . . . 209
8.10.14 clamp . . . 210
8.10.15 consgreet . . . 211
8.10.16 convert_string . . . 212
8.10.17 currentmap . . . 213
8.10.18 cvar_exists . . . 214
8.10.19 deletefile . . . 215
8.10.20 deleteproperty . . . 216
8.10.21 directmessage . . . 217
8.10.22 distance . . . 218
8.10.23 exec . . . 219
8.10.24 execclient . . . 220
8.10.25 execclient_all . . . 221
8.10.26 execute_command . . . 222
8.10.27 existproperty . . . 223
8.10.28 fileexists . . . 224
8.10.29 filesize . . . 225
8.10.30 fixed:f_abs . . . 226
8.10.31 fixed:f_arccos . . . 226
8.10.32 fixed:f_arccot . . . 227
8.10.33 fixed:f_arcosh . . . 227
8.10.34 fixed:f_arcoth . . . 228
8.10.35 fixed:f_arcsin . . . 228
8.10.36 fixed:f_arctan . . . 229
8.10.37 fixed:f_arctan_help . . . 229
8.10.38 fixed:f_arsinh . . . 230
8.10.39 fixed:f_artanh . . . 230
8.10.40 fixed:f_cos . . . 231
8.10.41 fixed:f_cosh . . . 231
8.10.42 fixed:f_cot . . . 232
8.10.43 fixed:f_coth . . . 232
8.10.44 fixed:f_degtorad . . . 233
8.10.45 fixed:f_euler . . . 233
8.10.46 fixed:f_faculty . . . 234
8.10.47 fixed:f_ln . . . 234
8.10.48 fixed:f_log10 . . . 235
8.10.49 fixed:f_logab . . . 236

8.10.50 fixed:f_max . . . 237
8.10.51 fixed:f_min . . . 237
8.10.52 fixed:f_pi . . . 238
8.10.53 fixed:f_power . . . 238
8.10.54 fixed:f_powere . . . 239
8.10.55 fixed:f_radtodeg . . . 239
8.10.56 fixed:f_sin . . . 240
8.10.57 fixed:f_sinh . . . 240
8.10.58 fixed:f_sqrt . . . 241
8.10.59 fixed:f_tan . . . 241
8.10.60 fixed:f_tanh . . . 242
8.10.61 fixed:fdiv . . . 242
8.10.62 fixed:ffract . . . 243
8.10.63 fixed:fixed . . . 243
8.10.64 fixed:fixedstr . . . 244
8.10.65 fixed:fmul . . . 245
8.10.66 fixed:strtofix . . . 246
8.10.67 fixtostr . . . 247
8.10.68 format_command . . . 248
8.10.69 fround . . . 249
8.10.70 funcidx . . . 250
8.10.71 get_serverinfo . . . 251
8.10.72 get_timer . . . 252
8.10.73 get_userArmor . . . 253
8.10.74 get_userAuthID . . . 254
8.10.75 get_userFrags . . . 255
8.10.76 get_userHealth . . . 256
8.10.77 get_userindex . . . 257
8.10.78 get_userinfo . . . 258
8.10.79 get_userIP . . . 259
8.10.80 get_username . . . 260
8.10.81 get_userorigin . . . 261
8.10.82 get_userSessionID . . . 262
8.10.83 get_userTeam . . . 263
8.10.84 get_userWONID . . . 264
8.10.85 get_vaultdata . . . 265
8.10.86 get_vaultnumdata . . . 266
8.10.87 getarg . . . 267
8.10.88 getchar . . . 267
8.10.89 getproperty . . . 268
8.10.90 getstring . . . 269
8.10.91 getstrvar . . . 270
8.10.92 gettarget . . . 271
8.10.93 getteamcount . . . 272

8.10.94 getvalue 272
8.10.95 getvar 273
8.10.96 glow 274
8.10.97 godmode 275
8.10.98 heapspace 276
8.10.99 help 277
8.10.100 index 278
8.10.101 kick 279
8.10.102 kill_timer 280
8.10.103 list_maps 281
8.10.104 listspawn 282
8.10.105 log 283
8.10.106 log_command 284
8.10.107 look_in_dir 285
8.10.108 maptime 286
8.10.109 matherror 287
8.10.110 max 288
8.10.111 maxplayercount 289
8.10.112 menu 290
8.10.113 message 291
8.10.114 messageex 292
8.10.115 min 293
8.10.116 motd 294
8.10.117 movespawn 295
8.10.118 nextmap 296
8.10.119 noclip 297
8.10.120 numargs 298
8.10.121 numtostr 299
8.10.122 playercount 300
8.10.123 playerinfo 301
8.10.124 playsound 302
8.10.125 plugin_checkcommand 303
8.10.126 plugin_command 304
8.10.127 plugin_connect 305
8.10.128 plugin_disconnect 306
8.10.129 plugin_exec 307
8.10.130 plugin_info 308
8.10.131 plugin_init 309
8.10.132 plugin_message 310
8.10.133 plugin_registercmd 311
8.10.134 plugin_registerhelp 312
8.10.135 plugin_registerinfo 313
8.10.136 pointto 314
8.10.137 print 314

8.10.138 printf . . . 315
8.10.139 rainbow . . . 315
8.10.140 random . . . 317
8.10.141 readfile . . . 318
8.10.142 reject_message . . . 319
8.10.143 reload . . . 320
8.10.144 removespawn . . . 321
8.10.145 resetfile . . . 322
8.10.146 rindex . . . 323
8.10.147 say . . . 324
8.10.148 say_command . . . 325
8.10.149 selfmessage . . . 326
8.10.150 servertime . . . 327
8.10.151 set_serverinfo . . . 329
8.10.152 set_timer . . . 330
8.10.153 set_vaultdata . . . 331
8.10.154 set_vaultnumdata . . . 332
8.10.155 setarg . . . 333
8.10.156 setproperty . . . 334
8.10.157 setstrvar . . . 335
8.10.158 slap . . . 336
8.10.159 slay . . . 337
8.10.160 snprintf . . . 338
8.10.161 spawn . . . 340
8.10.162 speakto . . . 341
8.10.163 strbreak . . . 342
8.10.164 strcasecmp . . . 343
8.10.165 strcasestr . . . 344
8.10.166 strcasestrx . . . 345
8.10.167 strcat . . . 346
8.10.168 strchr . . . 347
8.10.169 strcmp . . . 348
8.10.170 strcount . . . 349
8.10.171 strcpy . . . 350
8.10.172 strcspn . . . 351
8.10.173 streq . . . 352
8.10.174 strgsep . . . 353
8.10.175 strgsplit . . . 354
8.10.176 strgtok . . . 355
8.10.177 strgtokrest . . . 356
8.10.178 strinit . . . 356
8.10.179 strlen . . . 357
8.10.180 strmatch . . . 358
8.10.181 strncasecmp . . . 359

8.10.182 strncat 360
8.10.183 strncmp 361
8.10.184 strncpy 362
8.10.185 strpack 363
8.10.186 strrchr 364
8.10.187 strsep 365
8.10.188 strsplit 366
8.10.189 strspn 367
8.10.190 strstr 368
8.10.191 strstripquotes 369
8.10.192 strstrx 370
8.10.193 strsubst 371
8.10.194 strtok 372
8.10.195 strtokrest 373
8.10.196 strtonum 374
8.10.197 strtrim 375
8.10.198 strunpack 376
8.10.199 swapchars 377
8.10.200 systemtime 377
8.10.201 teleport 378
8.10.202 timeleft 379
8.10.203 tolower 380
8.10.204 toupper 380
8.10.205 typesay 381
8.10.206 unban 382
8.10.207 userlist 383
8.10.208 valid_map 384
8.10.209 valid_mapex 385
8.10.210 version 385
8.10.211 vote 386
8.10.212 vote_allowed 387
8.10.213 writefile 388
8.11 Compiler Fehler und Warnungen 389
8.12 Runtime Fehler 390
8.13 LogD Events 391
8.14 Properties in Small 393

A Besondere Customplugins 394
A.1 Plugin_sank_sounds 394
A.1.1 Konfiguration 394
A.1.2 Probleme 395
A.2 plugin_logd_ffmon 396
A.2.1 Voraussetzungen 397
A.2.2 Einstellungen 397

A.2.3 Probleme . . . 401

B Sonstiges **402**
B.1 Was macht eigentlich Setinfo? . . . 402
B.2 Interaktion zwischen HL Engine und GameDLL . . . 403
B.3 Res-Dateien . . . 404
B.4 HL-Sounds . . . 405
B.4.1 agrunt . . . 406
B.4.2 ambience . . . 406
B.4.3 apache . . . 407
B.4.4 aslave . . . 407
B.4.5 barnacle . . . 407
B.4.6 barney . . . 407
B.4.7 boid . . . 409
B.4.8 bullchicken . . . 409
B.4.9 buttons . . . 409
B.4.10 common . . . 410
B.4.11 controller . . . 410
B.4.12 debris . . . 410
B.4.13 doors . . . 411
B.4.14 fans . . . 411
B.4.15 fvox . . . 411
B.4.16 garg . . . 412
B.4.17 gman . . . 413
B.4.18 gonarch . . . 413
B.4.19 hassault . . . 413
B.4.20 headcrab . . . 413
B.4.21 hgrunt . . . 413
B.4.22 holo . . . 416
B.4.23 hornet . . . 416
B.4.24 houndeye . . . 417
B.4.25 ichy . . . 417
B.4.26 items . . . 417
B.4.27 leech . . . 417
B.4.28 nihilanth . . . 417
B.4.29 plats . . . 418
B.4.30 player . . . 418
B.4.31 roach . . . 419
B.4.32 scientist . . . 419
B.4.33 squeek . . . 422
B.4.34 tentacle . . . 422
B.4.35 tride . . . 422
B.4.36 turret . . . 422
B.4.37 vox . . . 422

B.4.38 weapons 427
B.4.39 x 428
B.4.40 zombie 428

C Changelog **429**

1. Einleitung

1.1. Vorwort

Natürlich gibt es das englische Handbuch, in dem eine nicht unbeträchtliche Menge an Information bereits vorhanden ist. Richtig zufrieden war ich aber mit dieser Dokumentation nicht wirklich. Mit der damals neu aufgesetzten deutschen Webseite www.adminmod.de[1] wollten wir den deutschen Nutzern eine verbesserte Dokumentation zukommen lassen. Die gesamte Beschreibung der Installation und Konfiguration wurde auf unseren HTML-Seiten dargestellt, was letztlich zu stark reduzierten Rückfragen im Forum führte.

Vielfach kam der Wunsch auf, dass wir doch das Ganze zum Download anbieten möchten. Aber wer hat schon Lust, die gesamte Dokumentation nochmal in ein anderes Format zu gießen. Dokumentation schreiben ist nämlich das nervigste an der gesamten Programmierarbeit. Ich experimentierte daher ein wenig mit LaTeX[2], um mir durch automatische Konvertierung die Arbeit zu erleichtern, ohne jedoch wesentliche Fortschritte zu machen. So blieb das Projekt erst einmal liegen.

Irgendwann holt einen dann auch noch das echte Leben (Real Life) ein, und man stellt fest, dass man für vieles, was einem bisher wichtig erschien, immer weniger Zeit aufwenden kann und will. Darüber hinaus schwindet allmählich das Interesse der Community an Admin Mod.

Da aber immer noch viele Leute Interesse an Admin Mod haben, wäre es meiner Meinung nach schade, wenn mein gesamtes Wissen einfach verschwinden würde. Vieles davon geht über das hinaus, was in den Anleitungen steht. So haben z.B. bislang wenige Leute Admin Mod mit einem Webforum verknüpft. Ich habe daher die Arbeit am Kompendium wieder aufgenommen.

Ich wollte die auf adminmod.de gesammelten Informationen in einer Art Buch aufbereiten. Die Texte auf adminmod.de waren inhaltlich sicherlich gut, aber die Ansprache des Lesers und der Aufbau ließen sich nicht auf ein Kompendium übertragen. Auch waren mir die Informationen nicht detailliert genug. Letztlich habe ich den Großteil neu geschrieben. Einiges wiederum wird man von der Webseite kennen.

Ich habe auch darüber nachgedacht, das Kompendium in Englisch zu schreiben. Aber hinsichtlich meiner begrenzten Zeit musste ich die Idee fallen lassen.

[1] http://www.adminmod.de
[2] http://www.latex-project.org

Ich hoffe, dass der eine oder andere mit diesem Kompendium etwas anfangen kann, oder sich eventuell auch inspirieren lässt.

Abschließend muss ich mich auch bei sechs Personen bedanken, da einige Ihrer Texte übernommen oder als Grundlage für dieses Kompendium genommen wurden. Dies sind:

- Sir Drink a lot
- Rinde
- Stillsetzhut
- Da Rope
- Biohazard
- Kleen13

1.2. Was ist Admin Mod?

Admin Mod[3] ist eines der bekanntesten Addons für den Half-Life Server[4] (bzw. dessen Modifikationen). Es gibt Admin Mod nur für die HL1-Engine nicht jedoch für die neuere Source-Engine. Eine Portierung ist mangels Programmierer leider gescheitert.

Admin Mod ist ein Metamod-Plugin, das fast universell mit jeder HL-Modifikation anwendbar ist, welche auch von Metamod[5] unterstützt wird. Admin Mod ist eine Open Source Software[6] deren Quellen bei Sourceforge[7] gehostet.

Warum eigentlich ein Addon für den Half-Life Server? Letztlich bietet auch der Server selber Administrationsmöglichkeiten an (rcon, votemap, etc.), nur sind diese recht eingeschränkt und schwierig zu bedienen. Die Serverengine bietet jedoch viel mehr und bessere Möglichkeiten an, die nicht per RCon genutzt werden können. Außerdem kann man die Zugriffsrechte deutlich feiner abstimmen.

Um die Administration zu erleichtern wurde daher Admin Mod entwickelt, was sich schnell zum beliebtesten Addon für HL-Server entwickelte.

Wichtige Features:

- Userrechte lassen sich im Gegensatz zu RCon feiner abstufen (Das Rcon-Passwort muss also nicht mehr weitergegeben werden)
- Pluginsystem (Das System lässt sich mittels Scripten (geschrieben in Small) leicht erweitern)

[3] http://www.adminmod.org
[4] http://store.steampowered.com/about
[5] http://www.metamod.org
[6] http://de.wikipedia.org/wiki/Open_Source
[7] http://sourceforge.net/projects/halflifeadmin

- Über 200 Plugins erhältlich
- Grafische Votesysteme
- Einfache Veränderung der Serverkonfiguration
- Reserviert Slots für Administratoren
- Unruhestifter können mit einer Vielzahl an Möglichkeiten zum Schweigen gebracht werden
- Fängt diverse Serverbugs ab
- u.v.m.

In Kombination mit LogD und StatsMe, weiteren Metamod-Plugins, können Scripte auch auf Serverevents wie Kills reagieren.

Admin Mod wird insbesondere wegen seiner Stabilität und Resourcenschonung immer noch eingesetzt, obwohl die Entwicklung am Plugin selber eingeschlafen ist. Die eigentliche Entwicklung hatte sowieso seit Einführung der Scripte vom Plugin weggeführt.

1.3. Warum nicht ein anderes Plugin nutzen?

Da es neben Admin Mod noch weitere Metamod-Plugins gibt, stellt sich die Frage, was die Unterschiede zwischen Admin Mod und beispielsweise AMXMod[8], AMXModX[9] oder ClanMod[10] sind?

Mit letzterem wird begonnen. Clanmod wurde von OLO entwickelt, um eine einfache Administration ohne viele Schnörkel zu ermöglichen. Lange Zeit war Clanmod das einzige Addon, das sich fast komplett über Menüs bedienen ließ. Leider war und ist die Erweiterung von Clanmod über das Gebotene hinaus schwer. Es gibt zwar inzwischen Möglichkeiten die Funktionen von Clanmod über Plugins zu erweitern, diese müssen jedoch in C++ geschrieben werden, und sind damit recht selten anzutreffen. Das Programm wurde von United Admins betreut, wird aber nicht mehr weitergepflegt.

Bei AMXMod(X) handelt es sich um eine ausgereifte und oft verwendete Software, die ihren Reiz aus den mannigfaltigen Funktionen und damit den unendlichen Möglichkeiten zum Eingreifen ins Spielgeschehen zieht. Gerade das, wie in Admin Mod verwendete, Small-Pluginsystem ermöglicht auf dem Gebiet der Fun-Plugins eine riesige Auswahl. Da AMXMod(X) eigentlich fast alle Admin Mod-Funktionen aufweist, können Admin Mod-Plugins mit relativ wenig Aufwand auf AMXMod(X) konvertiert werden.

Trotz der scheinbaren funktionalen Überlegenheit AMXMod(X)s hat Admin Mod immer noch seine Existenzberechtigung. Admin Mod mag nicht die Funktionsvielfalt besitzen,

[8] `http://amxmod.net`
[9] `http://www.amxmodx.org`
[10] `http://www.unitedadmins.com/index.php?p=content&content=clanmod`

wurde aber stets unter der Prämisse entwickelt, dass es ausschließlich zur Serveradministration verwendet werden sollte. Dem entsprechend sind Funktionen zur Beeinflussung des Gameplays verpönt. Das bedeutet, dass man auf einem Admin Mod-Server mehr oder weniger das Spiel vorfindet, was man erwartet. Unter AMXMod(X) können Plugins das Spielprinzip gravierend verändern. Auch Admin Mod bietet einige Funktionen wie Teleport oder Godmode, welche aber bei Anwendung stets allen Spielern mitgeteilt werden.

Ein weiterer Punkt ist der Schutz des Clients gegenüber der Willkür des Admins. Admin Mod erlaubt daher nicht das Ausführen aller Befehle beim Client. Es ist mir ein Rätsel, warum diverse Admins unbedingt die Konfiguration vermeintlicher Cheater zerstören müssen. Admin Mod bietet schon bezogen auf die Standardplugins genügend Möglichkeiten einen Spieler vom Server zu ekeln (und das mit Stil). Das Zerstören fremder Installationen ist eines Admins nicht würdig. Dies sollten sich die Nörgler in den Foren vor Augen führen. Aus diesem Grund werden die Restriktionen nicht gelockert.

Was Spielevents (z.B. Rundenstart, Kills) betrifft, so kann die Reaktion auf diese durch die Verwendung eines weiteren Metamod-Plugins namens LogDaemon (LogD[11] oder StatsMe[12] erweitert werden. Oftmals wird zudem übersehen, dass auch Admin Mod Menüs besitzt, die in vielen Customplugins eingesetzt werden.

Es ist inzwischen möglich viele nichtadminitrative Plugins zu verwenden, auch wenn bei der Erstellung nie daran gedacht wurde.

Clanmod ist sicherlich vorzugsweise etwas für den ClanWar-Server, AMXMod(X) etwas für den Publicserver bei Admins, die für Abwechslung sorgen möchten. Admin Mod hingegen kann für ClanWar-Server als auch Publicserver eingesetzt werden. Admin Mod kann dabei fast Clanmod ersetzen, bietet jedoch die Sicherheit auf Publicservern, dass alles so funktioniert, wie es das Gameplay des Mods mal vorgesehen hat.

Ich persönlich versuche AMXMod-Server zu vermeiden, aber das ist mittlerweile schwer geworden.

[11] http://logd.sf.net/
[12] http://www.unitedadmins.com/index.php?p=content&content=statsme

2. Historie

Es ist nicht einfach etwas über Admin Mods Historie aus den Weiten des Netzes zu erfahren. Vieles blieb wohl mehr oder weniger undokumentiert. Dennoch soll an dieser Stelle versucht werden, die vorhandenen Informationen zu einem groben Gesamtbild zusammenzufassen.

Der Vater von Admin Mod ist Alfred Reynolds (inzwischen bei Valve[1] beschäftigt). Anfang 2000 betreute er einige Server, auf denen er nicht ständig als Administrator unterwegs sein konnte. So musste er das RCon-Passwort an andere Leute weitergeben, die dies teilweise schamlos ausnutzten. Inspirieren ließ er sich durch ein Admin-Plugin für die Quake2-Engine (BW-Admin[2]). Mit der einführenden Hilfe von Jeffrey „botman“ Broome[3] begann er mit der HL SDK den ersten Code im Februar 2000 für Admin Mod zu schreiben.

Das Projekt nahm schnell Fahrt auf. Auf Grund der häufigen Updates wurde Alfred sogar als „Release-a-Day Man“ bezeichnet. Im März 2000 wurde das Projekt auf Sourceforge umgezogen, und mit der Version 0.72 das erste offizielle Release freigegeben. Im gleichen Monat wurde bereits durch einen User der Code für den MySQL-Zugriff in Admin Mod aufgenommen. Zu diesem Zeitpunkt gab es noch kein Metamod, so dass Admin Mod stets an die sich laufend ändernden Mods angepasst werden musste. Dies nahm sehr viel Zeit in Anspruch. Anfang Mai 2000 wurde mit der Version 0.84 der Release der Version 1.0 eingeleitet, die dann noch weitere Mods unterstützen sollte. Im Juli 2000 war es soweit, dass die Version 1.0 veröffentlicht wurde.

Gleichzeitig mit der Veröffentlichungsphase der Version 1 wurde bereits mit den Arbeiten an Version 2 begonnen und wenige Tage nach Version 1 veröffentlicht. Die große Neuerung war der Scripting-Support (Small), der sich jedoch auf ein einzelnes, großes File beschränkte. Diese Version wurde noch bis Anfang Dezember 2000 weiterentwickelt. Einige Mods kamen hinzu, und es wurde an den Scripting-Funktionen gefeilt. Während dieser Zeit, im Oktober 2000, wurde auch die deutsche Web-Dependence (adminmod.de) eröffnet. Außerdem wuchs das Team um Alfred zunehmend, wobei es sich eigentlich mehr um Interessierte handelte als ein richtiges Team.

Ende März 2001 wurde dann die Version 2.50 veröffentlicht, die den letzten großen Meilenstein in der Admin Mod Entwicklung darstellt. Admin Mod ist seit diesem Zeitpunkt ein Metamod-Plugin. Neu eingeführt wurde das Pluginsystem, das weg von der großen,

[1] http://www.valvesoftware.com
[2] http://bw-admin.sourceforge.net
[3] http://botman.planethalflife.gamespy.com/about_me.shtml

schwierig zu erweiternden Scripting-Datei hin zu einem einfachen, modularen System führte. Trotz der damals unbekannten Sprache Small entwickelte sich eine große Community rund um das Thema Scripting, die wesentlich zum Erfolg Admin Mods beitrug.

Die Entwicklung Admin Mods zielte im Weiteren auf die Erweiterung der Funktionen des Scriptingsystems, Abrunden der bestehenden Funktionalität und Bugfixes. In 2002 kam es zu einem größeren Umbruch bei Admin Mod. Valve konnte Alfred als Mitarbeiter gewinnen, was seine Zeit für Admin Mod jedoch stark einschränkte. So übernahm Da Rope von diesem Zeitpunkt an immer mehr die Funktion des Projektmanagers.

Mit der erweiterbaren Funktionalität wuchs auch das Interesse an weitgehenderen Eingriffen ins Spiel. Während sich zunächst LogDaemon und dann StatsMe als Event-Erkennung noch um Interaktion mit Admin Mod bemühten, wurden mit Clanmod und AMXMod eigenständige Metamodplugins entwickelt, die eine ähnlich Funktionalität aufwiesen wie Admin Mod. Hierbei wurde aber auch schon tiefgreifend ins Spielgeschehen eingegriffen, was hingegen bei der Admin Mod Entwicklung bewusst vermieden und unterbunden wurde.

Im August 2004 wurde die bislang letzte Admin Mod Version veröffentlicht. Einige Wiederbelebungsversuche, z.B. die Portierung auf die Source-Engine oder eine eigenes Eventsystem, verliefen im Sande, so dass die Entwicklung sich auf den Scriptingbereich beschränkte. Nachdem AMXMod und AMXModX inzwischen Admin Mod den Rang abgelaufen haben, sind im Wesentlichen nur noch zwei Personen im Support von Admin Mod übrig geblieben (Sir Drink a lot und ich).

Admin Mod wird zwar heute (2009) nicht mehr oft benutzt, ist aber weiterhin voll funktionstüchtig. Resourcenschonung und die eingeschränkten Eingriffsmöglichkeiten in die Spielphysik sind ein wesentliches Plus, das oft übersehen wird.

3. Installation

3.1. Voraussetzungen

Es wird vorausgesetzt, dass bereits ein lauffähiger Half-Life Server[1] zur Verfügung steht. Dabei sollte sicher gestellt werden, dass die neuste Version des Servers (HL1) sowie der eingesetzten Modifikation (z.B. Counter-Strike) installiert ist. Es kann damit ausgeschlossen werden, dass Admin Mod fälschlicherweise für etwaige bestehende Probleme verantwortlich gemacht wird. Das erspart Ärger und peinliche Fragen im Forum[2].

Es sollte nun die aktuelle Admin Mod-Version (2.50.60[3]) und am besten die Metamod-Version 1.19[4] heruntergeladen werden, da die bei Admin Mod beigelegte Metamod-Version veraltet ist und ggf. zum Serverabsturz führen kann.

Die gesamte Zip-Datei (Windows) oder TGZ-Datei (Linux) ist in ein beliebiges Verzeichnis zu entpacken. Im eigenen Interesse sollte aber nicht in die Serververzeichnisse entpackt werden. Es handelt sich um eine Distribution, die zunächst erst installiert werden muss. Das Entpacken in die Server-Verzeichnisse wird nur zur Konfusion führen.

Es gibt nun die Möglichkeit ein Skript die Installation übernehmen zu lassen. Unter Windows führt man die install_admin.vbs und unter Linux die install_admin aus und folgt den Anweisungen. Dies funktioniert ausschließlich, wenn ein Zugriff über SSH oder Remote Control zur Verfügung steht. Mit einem FTP-Zugang ist dies nicht möglich!

Als zweite Möglichkeit bietet sich die manuelle Installation Admin Mods an. Diese funktioniert auch mit einem FTP-Zugang. Es ist in jedem Fall zu empfehlen, eine manuelle Installation vorzunehmen, da es das Verständnis für Admin Mod verbessert. Im Weiteren wird aus diesem Grund nicht weiter auf die automatische Installation eingegangen.

3.2. Installation (Windows)

Wie bereits erwähnt ist die Distributionsstruktur nicht identisch mit der Installationsstruktur auf dem Server. Die Dateien müssen erst an die richtige Stelle kopiert werden. Diese Anleitung erklärt, welche Dateien aus welchen Verzeichnissen der Distribution in

[1] http://store.steampowered.com/about/
[2] http://forum.adminmod.de/
[3] http://sourceforge.net/project/showfiles.php?group_id=3570&package_id=3527
[4] http://sourceforge.net/project/showfiles.php?group_id=24195&package_id=119653

welche Verzeichnisse auf dem Server kommen. Sofern es nicht schon existiert, ist auf dem Server ein Verzeichnis „addons\adminmod" im gewünschten Mod-Verzeichnis (z.B. cstrike[5]) zu erstellen. Es ist zu beachten, dass sich die Verzeichnisnamen zwischen der alten HL 4.1.1.x Version gegenüber Steam unterscheiden:

4.1.1.x: z.B. `X:\Hlserver\cstrike`
Steam: z.B. `X:\Steam\SteamApps\me@email.com\dedicated server\cstrike`

Bei einem Listenserver (man spielt selbst auf dem gleichen Rechner mit) muss natürlich das Spielverzeichnis gewählt werden

```
4.1.1.x: z.B. X:\Half-Life\cstrike
Steam: X:\Steam\SteamApps\me@email.com\counter-strike\cstrike
```

Im Mod-Verzeichnis werden 3 weitere Verzeichnisse benötigt (config, dlls und scripts).

```
Erstelle addons\adminmod\config
Erstelle addons\adminmod\dlls
Erstelle addons\adminmod\scripts
```

Aus der entpackten Admin Mod-Distribution können alle Dateien aus dem Verzeichnis „scripting\binaries" in das „addons\adminmod\scripts" Verzeichnis auf dem Server kopiert werden. Außerdem empfiehlt es sich, die plugin.ini und die adminmod.cfg aus dem „config" Verzeichnis in das gleichnamige Verzeichnis auf dem Server zu kopieren. Die plugin.ini gibt an, wo Admin Mod die auf dem Server installierten Plugins findet, während die adminmod.cfg die grundlegende Konfiguration von Admin Mod beinhaltet. Ferner sollte auch eine leere users.ini hierhin geschoben werden. Also:

```
Kopiere scripting\binaries\*.amx ⟶ addons\adminmod\scripts
Kopiere config\plugin.ini ⟶ addons\adminmod\config
Kopiere config\adminmod.cfg ⟶ addons\adminmod\config
Erstelle addons\adminmod\config\users.ini
```

Nun muss noch Metamod[6] installiert werden. Dazu wird wieder im Mod-Verzeichnis ein Unterverzeichnis „addons\metamod" angelegt. Hier bedarf es eines „dlls" Directories, in welches die metamod.dll kopiert wird. Es kann außerdem schonmal eine leere „plugins.ini" (Unbedingt das „s" am Ende beachten) in „addons\metamod" kopiert werden (einfach eine leere Textdatei in plugins.ini umbenennen).

```
Erstelle addons\metamod
Erstelle addons\metamod\dlls
Kopiere dlls\metamod.dll ⟶ addons\metamod\dlls
Erstelle addons\metamod\plugins.ini
```

Soll Admin Mod auf einem Dedicated Server installiert werden, so ist der Eintrag „exec addons/adminmod/config/adminmod.cfg" am Ende der server.cfg einzufügen. Bei einem Listenserver muss dieser Eintrag natürlich stattdessen in die listenserver.cfg eingefügt

[5] http://www.counter-strike.net/
[6] http://www.metamod.org/

werden. Somit werden beim Serverstart die grundlegenden Einstellungen von Admin Mod geladen.

Folgende Dateien sind zu kopieren:

`Kopiere dlls\admin_MM.dll` ⟶ `addons\adminmod\dlls`
`Kopiere dlls\amx_admin.dll` ⟶ `addons\adminmod\dlls`

Folgender Eintrag gehört außerdem in die addons\metamod\plugins.ini:

```
win32 addons/adminmod/dlls/admin_MM.dll
```

Nun noch die letzte Datei die liblist.gam (ist im „Mod“-Dir auf dem Server). Die Datei öffnet man in einem Texteditor und ändert die Zeile:

```
gamedll "dlls\mp.dll" zu:
gamedll "addons\metamod\dlls\metamod.dll"
```

Wahlweise ist es mit einem neueren Steamserver auch möglich Metamod direkt über einen Startparameter zu starten. Dies umgeht auch das ständige Überschreiben der liblist.gam beim Serverupdate.

```
-dll addons\metamod\dlls\metamod.dll
```

Damit ist die Installation abgeschlossen und ein Serverrestart notwendig. Ob alles funktioniert ist via Serverconsole oder einem Remotetool (z.B. HLSW[7]) zu erkennen. Durch Eingabe von „meta list“ kann überprüft werden, ob Metamod geladen wurde, bzw. ob es in der Lage war Admin Mod zu starten. Durch den Befehl „admin_cmd admin_version“ kann dann kontrolliert werden, ob Admin Mod ordnungsgemäß funktioniert. Bei Problemen sollte im Modverzeichnis eine autoexec.cfg erstellen, in der die Zeile „log on“ steht. Anschließend können die Logdateien im neuen „logs“ Verzeichnis Hinweise geben, warum etwas nicht geht.

Einen Überblick über die entstandene Verzeichnisstruktur ist dem Kapitel 3.5 zu entnehmen.

3.3. Installation (Linux/FreeBSD)

Wie bereits erwähnt ist die Distributionsstruktur nicht identisch mit der Installationsstruktur auf dem Server. Die Dateien müssen erst an die richtige Stelle kopiert werden. Diese Anleitung erklärt, welche Dateien aus welchen Verzeichnissen der Distribution in welche Verzeichnisse auf dem Server kommen. Sofern es nicht schon existiert, ist auf dem Server ein Verzeichnis „addons/adminmod“ im gewünschten Mod-Verzeichnis (z.B. cstrike[8]) zu erstellen. Hier werden drei weitere Verzeichnisse benötigt (config, dlls und scripts).

[7] http://www.hlsw.de/
[8] http://www.counter-strike.net/

```
Erstelle addons/adminmod/config
Erstelle addons/adminmod/dlls
Erstelle addons/adminmod/scripts
```

Aus der entpackten Admin Mod-Distribution können alle Dateien aus dem Verzeichnis „scripting/binaries“ in das „addons/adminmod/scripts“ Verzeichnis auf dem Server kopiert werden. Außerdem empfiehlt es sich, die plugin.ini und die adminmod.cfg aus dem „config“ Verzeichnis in das gleichnamige Verzeichnis auf dem Server zu kopieren. Die plugin.ini gibt an, wo Admin Mod die auf dem Server installierten Plugins findet, während die adminmod.cfg die grundlegende Konfiguration von Admin Mod beinhaltet. Ferner sollte auch eine leere users.ini hierhin geschoben werden. Also:

```
Kopiere scripting/binaries/*.amx ⟶ addons/adminmod/scripts
Kopiere config/plugin.ini ⟶ addons/adminmod/config
Kopiere config/adminmod.cfg ⟶ addons/adminmod/config
Erstelle addons/adminmod/config/users.ini
```

Nun muss noch Metamod[9] installiert werden. Dazu wird wieder im Mod-Verzeichnis ein Unterverzeichnis „addons/metamod“ angelegt. Hier bedarf es eines „dlls“ Directories, in welches die metamod.dll kopiert wird. Es kann außerdem schonmal eine leere „plugins.ini“ (Unbedingt das „s“ am Ende beachten) in „addons/metamod“ kopiert werden (einfach eine leere Textdatei in plugins.ini umbenennen).

```
Erstelle addons/metamod
Erstelle addons/metamod/dlls
Kopiere dlls/metamod_i386.so ⟶ addons/metamod/dlls
Erstelle addons/metamod/plugins.ini
```

Bei der 64 bit Version lautet die Datei metamod_amd64.so.

Im Mod-Verzeichnis ist der Eintrag „exec addons/adminmod/config/adminmod.cfg“ am Ende der server.cfg einzufügen. Somit werden beim Serverstart die grundlegenden Einstellungen von Admin Mod geladen.

Folgende Dateien sind zu kopieren:

```
Kopiere dlls/admin_MM_i386.so ⟶ addons/adminmod/dlls
Kopiere dlls/amx_admin.so ⟶ addons/adminmod/dlls
```

Für die 64 bit Version müssen folgende Dateien kopiert werden:

```
Kopiere dlls/admin_MM_amd64.so ⟶ addons/adminmod/dlls
Kopiere dlls/amx_admin_amd64.so ⟶ addons/adminmod/dlls
```

Folgender Eintrag gehört außerdem in die addons/metamod/plugins.ini:

```
linux addons/adminmod/dlls/admin_MM_i386.so
```

Für die 64 bit-Linuxversion, lautet die Zeile folgendermaßen:

[9] http://www.metamod.org/

`lin64 addons/adminmod/dlls/admin_MM_amd64.so`

Nun noch die letzte Datei die liblist.gam (ist im „Mod"-Dir auf dem Server). Die Datei öffnet man in einem Texteditor und ändert die Zeile:

`gamedll_linux "dlls/cstrike_i386.so"` in:
`gamedll_linux "addons/metamod/dlls/metamod_i386.so"` oder
`gamedll_linux "addons/metamod/dlls/metamod_amd64.so"` für 64 bit

Wahlweise ist es mit einem neueren Steamserver auch möglich Metamod direkt über einen Startparameter zu starten. Dies umgeht auch das ständige Überschreiben der liblist.gam beim Serverupdate.

`-dll addons/metamod/dlls/metamod_i386.so` bzw.
`-dll addons/metamod/dlls/metamod_amd64.so`

Damit ist die Installation abgeschlossen und ein Serverrestart notwendig. Ob alles funktioniert ist via Serverconsole oder einem Remotetool (z.B. HLSW[10]) zu erkennen. Durch Eingabe von „meta list" kann überprüft werden, ob Metamod geladen wurde, bzw. ob es in der Lage war Admin Mod zu starten. Durch den Befehl „admin_cmd admin_version" kann dann kontrolliert werden, ob Admin Mod ordnungsgemäß funktioniert. Bei Problemen sollte im Modverzeichnis eine autoexec.cfg erstellen, in der die Zeile „log on" steht. Anschließend können die Logdateien im neuen „logs" Verzeichnis Hinweise geben, warum etwas nicht geht.

Es ist unbedingt zu beachten, dass Linux und FreeBSD Groß- und Kleinschreibung unterscheiden. Die Datei- und Verzeichniseinträge sind ungültig, wenn die Schreibweise des Eintrags von der eigentlichen Datei bzw. des Verzeichnisses abweicht!

Einen Überblick über die entstandene Verzeichnisstruktur ist dem Kapitel 3.5 am Beispiel der Windows-Installation zu entnehmen. Abgesehen von der DLL-Bezeichnung ist sie aber identisch mit der Linux-Installation.

3.4. MySQL/PostgreSQL Installation

Um große Usergruppen sinnvoll verwalten zu können, bietet Admin Mod eine MySQL-[11] und eine PostgreSQL-Unterstützung[12] an. Es ist ebenso möglich Admin Mod in bestehende Content Management Systeme (CMS) oder Foren einzubauen. Steht der Gameserver und der SQL-Server bereits, ist es recht einfach Admin Mod-SQL aufzusetzen. Ob der Aufwand den Zweck allerdings rechtfertigt, muss jeder für sich entscheiden. Insbesondere wenn es nur um einen Clanserver mit vielleicht 20 Usern geht, sollte man sich mit den Dateien begnügen. Der Aufwand steht hier in keinem Bezug zum Nutzen.

Für die Umsetzung sollte auf dem Gameserver zunächst Admin Mod ohne SQL-Un-

[10] http://www.hlsw.de/
[11] http://www.mysql.com/
[12] http://www.postgresql.org/

terstützung (Windows, Linux) laufen. Dies ermöglicht zum einen Probleme schneller aufzufinden und zum anderen ein Fallbacksystem für den Fall des Verbindungsverlustes zum SQL-Server zu haben.

Man sollte sich außerdem Gedanken darüber machen, ob man den SQL-Server auf dem Gameserver oder einem externen Rechner laufen lässt. Dies hat beides Vor- und Nachteile. Während der SQL-Server auf dem Gameserver keine Verbindungsverluste hat, produziert er aber zusätzliche Prozessorlast. Dies wird nochmals ungemein verstärkt, wenn zur Datenadministration ein Webserver herhalten soll. Ein SQL-Server außerhalb des eigentlichen Gameservers kann erheblich zur Lastverteilung beitragen, hat jedoch eine höhere Latenzzeit bei Queries (evtl. Lags) und läuft Gefahr die Verbindung komplett zu verlieren. Der SQL-Server sollte also netzwerktechnisch in direkter Umgebung des eigentlichen Servers stehen. Außerdem muss man einen Port für SQL nach außen öffnen, was sicherheitstechnisch auch nicht besonders zu empfehlen ist.

Die größte Hürde für die meisten Anwender besteht jedoch darin, dass es (meines Wissens) kein wirklich gutes Frontend für die Administration gibt. Man wird also z.B. seine eigenen PHP-Kenntnisse bemühen und sich ein eigenes System basteln müssen. Zur Not tut es aber auch PHPMyAdmin[13] oder phpPGAdmin[14].

Läuft der Gameserver mit Admin Mod einwandfrei, kann man sich die SQL-Version besorgen und einspielen[15]. Die SQL-Version beinhaltet nur die .dll oder .so Datei. Die restlichen Dateien sind der normalen Distribution zu entnehmen (Noch ein Grund mehr, zunächst die normale Admin Mod-Version zu installieren!).

Windows:
`Kopiere AdminSQL\admin_MM_mysql.dll → addons\adminmod\dlls (MySQL)`
`Kopiere AdminSQL\admin_MM_pgsql.dll → addons\adminmod\dlls (PGSQL)`

Linux:
`Kopiere AdminSQL/admin_MM_i386_mysql.so → addons/adminmod/dlls (MySQL)`
`Kopiere AdminSQL/admin_MM_i386_pgsql.so → addons/adminmod/dlls (PGSQL)`

Linux 64-bit:
`Kopiere AdminSQL/admin_MM_amd64_mysql.so → addons/adminmod/dlls (MySQL)`
`Kopiere AdminSQL/admin_MM_amd64_pgsql.so → addons/adminmod/dlls (PGSQL)`

In der Linuxversion liegen noch zwei zusätzliche Bibliotheken bei, die in das Verzeichnis mit hlds_run („Gameserver-Root“) kopiert werden sollten, wenn etwas wider erwarten nicht funktioniert.

`Kopiere AdminSQL/libmysqlclient.so.10 ⟶ hlds_l (MySQL)`
`Kopiere AdminSQL/libpg.so.3 ⟶ hlds_l (PostgreSQL)`

Anschließend muss noch die plugins.ini von Metamod geändert werden. Nach dem Neustart sollte der Gameserver genau so laufen wie vorher. Es wird nur noch nicht auf den

[13] http://www.phpmyadmin.net/
[14] http://phppgadmin.sourceforge.net/
[15] https://sourceforge.net/project/showfiles.php?group_id=3570&package_id=3991

SQL-Server zugegriffen, sondern es läuft die Fallback-Konfiguration. Wie die Tabellen zu erstellen sind bzw. welche Einstellungen zu beachten sind, geht aus der Konfigurationsanweisung für MySQL und PostgreSQL hervor.

3.5. Verzeichnisstruktur

Anhand der Windows-Installation von Counter-Strike ist in Abbildung 3.1 beispielhaft die Verzeichnisstruktur dargestellt. Die fett gedruckten Dateien und Verzeichnisse stellen sozusagen die Minimalinstallation von Admin Mod dar. Dabei heißt minimal jedoch nicht, dass man nicht noch mit weniger auskommt. Einige Dinge wie plugin_antiflood sind jedoch für den sicheren Serverbetrieb sehr zu empfehlen und sollten nicht entfernt werden.

3.6. Admin Mod mit Bots

Einige Bots sind nicht als Metamodplugin erhältlich, so dass sich eine parallele Installation mit Admin Mod nicht als so einfach erweist. In diesem Fall muss man Metamod mitteilen, dass es die Daten nicht an die Gameengine sondern an den Bot durchschleift. Dies geschieht über den Startparameter „localinfo mm_gamedll“.

Windows:
```
C:\HLServer\hlds.exe -game cstrike +localinfo mm_gamedll
addons/meinbot/meinbot.dll +map de_dust
```

Linux:
```
./hlds_run -game cstrike +localinfo mm_gamedll addons/meinbot/meinbot.so
+map de_dust
```

Alternativ dazu kann man auch im Verzeichnis addons/metamod eine Konfigurationsdatei namens config.ini erstellen. Der Inhalt müsste dann folgendermaßen aussehen:

Windows:
```
gamedll addons/meinbot/meinbot.dll
```

Linux:
```
gamedll addons/meinbot/meinbot.so
```

Auf die Verwendung von „localinfo mm_gamedll“ als Startparameter kann dann verzichtet werden.

Je nach Bot muss man ausprobieren, ob man admin_bot_protection in der Konfigurationsdatei adminmod.cfg auf 1 gesetzt haben muss, um einen Serverabsturz zu vermeiden.

Eine Abhandlung über die Interaktion zwischen Metamod, Bot und HL ist im Anhang von daRope dargelegt.

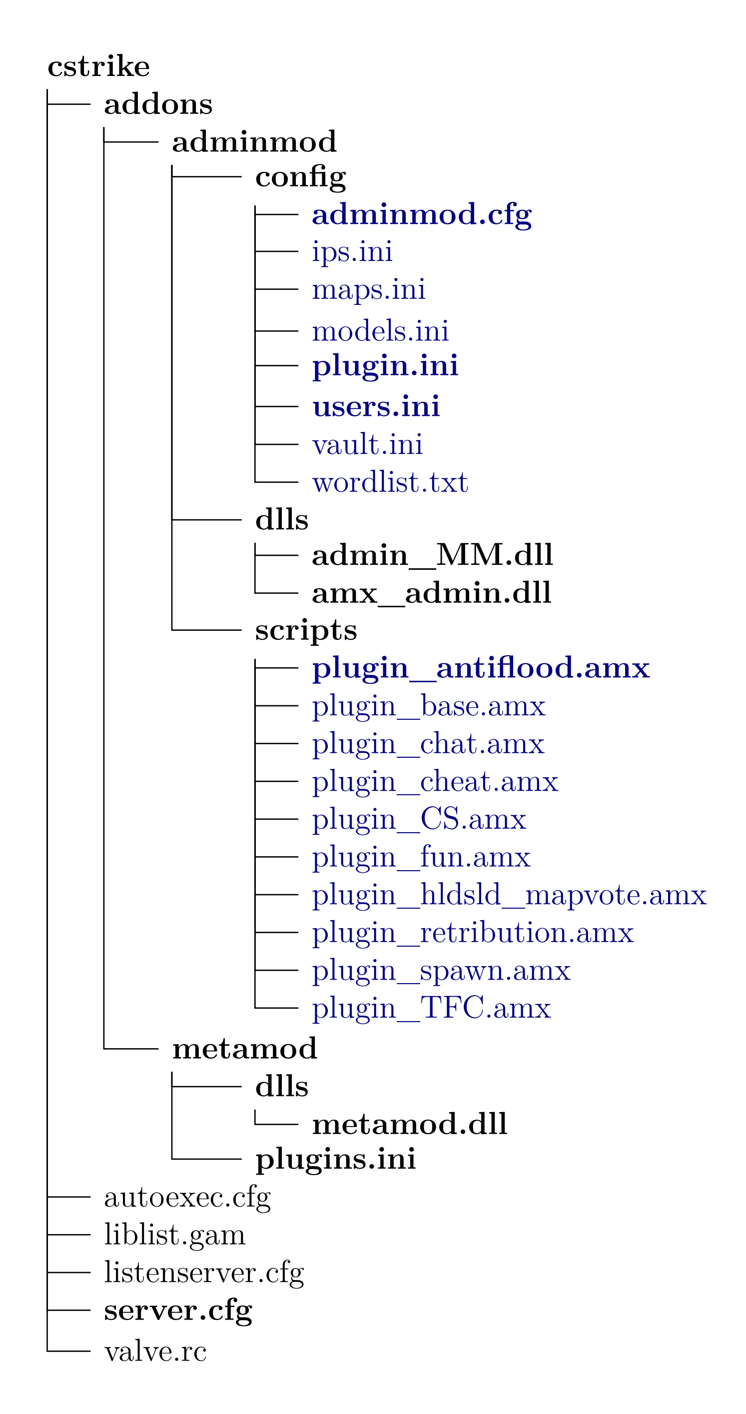


Abbildung 3.1.: Verzeichnisstruktur einer Windows-Installation

4. Konfiguration

4.1. Admin Mod einrichten (adminmod.cfg)

Dieser Abschnitt beschäftigt sich mit den Grundeinstellungen von Adminmod. Diese findet man in der adminmod.cfg („addons/adminmod/config“). Sie sollte stets aus der server.cfg heraus ausgeführt werden („exec addons/adminmod/config/adminmod.cfg“). Im Weiteren sollen die einzelnen Einstellungen (CVars) vorgestellt werden. Variablen, die nicht in der Standard adminmod.cfg stehen, können einfach ergänzt werden.

Änderungen an der adminmod.cfg werden nicht automatisch ausgeführt. In der Grundeinstellung des Half-Life Servers wird die Datei nur beim Starten des Servers geladen, da auch die server.cfg nur zu diesem Zeitpunkt geladen wird. Nur wenn man die Variable mapchangecfgfile auf „server.cfg“ oder „addons/adminmod/config/adminmod.cfg“ stellt, werden die Änderungen beim Mapwechsel übernommen. Alternativ kann man nach einer Änderung auch einfach „exec addons/adminmod/config/adminmod.cfg“ in der Serverconsole eingeben.

4.1.1. admin_balance_teams

```
admin_balance_teams <#>
```

Diese Variable wird vom TFC-Plugin verwendet. Bei einer zahlenmäßigen Überlegenheit des gegnerischen Teams wird diese automatisch ausgeglichen, indem einige Spieler ins gegnerische Team verschoben werden. Die Zahl hinter der CVar gibt den Überhang an Spielern an, ab der das Plugin aktiv wird. Die Standardeinstellung ist 0 (deaktiviert). Für CS hat diese Variable auf Grund der bereits integrierten Lösung keine Bedeutung.

Beispiel:

```
admin_balance_teams 3
```

Sobald ein Team 3 Spieler mehr hat als das andere, werden die Teams ausgeglichen.

4.1.2. admin_bot_protection

`admin_bot_protection <#>`

Hat man Bots auf seinem Server, so sollte man ggf. die Variable auf 1 setzen. Einige Bots vertragen keine Serverbefehle wie beispielweise „kick“. Diese Einstellung verhindert, dass ein solcher Befehl auf den Bot ausgeführt werden kann. Man läuft also Gefahr, einen Servercrash zu erleiden. Einfach ausprobieren, ob der Bot stabil mit der eigenen Admin Mod Installation läuft. Standardmäßig ist die 0 voreingestellt (deaktiviert).

Beispiel:
`admin_bot_protection 1`

Die Botprotection ist mit dieser Einstellung aktiviert.

4.1.3. admin_connect_msg

`admin_connect_msg "<string>"`

Diese Meldung wird dem jeweiligen Spieler nach dem Spieleinstieg auf den Server angezeigt (plugin_message).

Mit `admin_connect_msg 0` lässt sich die Nachricht komplett abschalten.

Beispiel:
`admin_connect_msg "Welcome to the pleasuredome..."`

60 Sekunden nach dem Spieleinstieg wird in der Mitte des Bildschirms „Welcome to the pleasuredome...“ angezeigt.

Siehe auch:
admin_repeat_msg, admin_repeat_freq

4.1.4. admin_cs_restrict

`admin_cs_restrict <#>`

Diese Variable wird vom CS-Plugin verwendet. Wenn auf 1 gesetzt, ist es möglich den Kauf von bestimmten Waffen (z.B. AWM oder Schild) oder Equipment zu verbieten. Es ist 0 voreingestellt.

Beispiel:
`admin_cs_restrict 1`

Das Verbieten von Waffen und Equipment ist damit aktiviert.

Siehe auch:
admin_repeat_msg, admin_repeat_freq

4.1.5. admin_debug

`admin_debug <#>`

Eigentlich ist diese Variable nur für Scripter interessant, die ihre Plugins debuggen wollen. Für den Normalbetrieb ist es nicht zu empfehlen, daher sollte der Wert auf 0 belassen werden. Für das Debuggen kann man den Wert von 1 bis 4 erhöhen. Damit erhöht sich auch die Informationstiefe und nicht unerheblich die Logfilegröße!! Mehr Informationen zum Spielerconnect erhält man hingegen mit admin_devel.

Beispiel:
`admin_debug 4`

Admin Mod lässt sich mit dieser Einstellung tief in die Karten schauen. Der erfahrene Scripter kann mit den geloggten Ausgaben meist schnell erkennen, wo der Bug im Plugin liegt. Standard: 0

Siehe auch:
amv_log_passwords, admin_devel

4.1.6. admin_devel

`admin_devel <#>`

Diese Variable gibt einige zumeist grundlegende Informationen zum Spielerconnect in den Logs aus. Außer für Programmierer an der Admin Mod Bibliothek ist diese Variable uninteressant und sollte auf der Standardeinstellung 0 belassen werden. Mögliche Werte sind 0 bis 3 (wie bei admin_debug: je größer die Zahl, desto mehr Information).

Beispiel:
`admin_devel 3`

Gibt alle Informationen zum Spielerconnect aus.

Siehe auch:
admin_debug

4.1.7. admin_fun_mode

`admin_fun_mode <#>`

Man kann mit dieser Variable erlauben den Glow- bzw. Disco-Modus zu verwenden.

Wird für das Fun-Plugin benötigt. Die Variable kann man auch temporär mit dem Befehl admin_fun verstellen. Die Glow-Funktion möchten die meisten Serverbetreiber nicht missen, daher sollte diese Einstellung auf 1 gestellt werden. Standard: 0

Beispiel:
`admin_fun_mode 1`

Der Gloweffekt wird mit dieser Einstellung freigeschaltet.

Siehe auch:
admin_fun

4.1.8. admin_fx

`admin_fx <#>`

Habt Ihr Euch schon immer gefragt, warum auf anderen Servern geslayte Spieler mit einem Donnerschlag explodieren? Setzt mal diese Variable auf 1. Dann tun sie das auch auf Eurem Server. Standard: 0

Beispiel:
`admin_fx 1`

Diese Einstellung muss vorgenommen werden, damit die „Nachricht“ des Admins stilvoll unterstrichen wird.

4.1.9. admin_gag_name

`admin_gag_name <#>`

De-/aktiviert die Möglichkeit zur Veränderung des eigenen Namens, wenn man geknebelt (gag) wurde. Einige Spieler versuchen über die Änderung des Namens trotz Knebelung weiter zu kommunizieren, indem sie ihre Nachrichten in ihren Namen schreiben. (plugin_retribution) Voreinstellung: 0

Beispiel:
`admin_gag_name 1`

Diese Einstellung verhindert, dass ein geknebelter Spieler über seinen Namen weiterkommunizieren kann.

Siehe auch:
admin_gag_sayteam

4.1.10. admin_gag_sayteam

`admin_gag_sayteam <#>`

Wenn auf 1 gesetzt, verhindert diese Variable, dass geknebelte Spieler den Teamchat benutzen können. (plugin_retribution) Die Voreinstellung ist 0.

Beispiel:
`admin_gag_sayteam 1`

Ein geknebelter Spieler kann neben dem allgemeinen Chat auch den Teamchat nicht mehr benutzen.

Siehe auch:
admin_gag_name

4.1.11. admin_highlander

`admin_highlander <#>`

Es kann nur einen geben! Ja richtig, es gibt bei 1 immer nur EINEN Admin auf dem Server und zwar den mit den mit der höchsten Rechtesumme.

Ein "kleinerer" Admin fällt auf das in default_access festgelegte Level ab, sobald ein "höherer" Admin auf den Server kommt.

Der Hauptadmin soll natürlich über allen stehen. Trotzdem möchte man die Rechte einiger Admins nicht beschneiden und räumt Ihnen den selben Accesslevel ein. Dann wird es mit der Highländervergabe problematisch. Eine einfache Lösung bietet sich mit dem nichtgenutzten Accesslevel 131072 an. Dieses addiert man zum Accesslevel des Hauptadmins hinzu. Anschließend ist der Hauptadmin immer Highlander, die Rechte 65336 stehen aber auch den Unteradmins zur Verfügung, während der Hauptadmin offline ist. Default: 0

Beispiel:
`admin_highlander 1`

Mit dieser Einstellung wird derjenige alleiniger Admin, der den höchsten Accesslevel besitzt.

Siehe auch:
admin_ignore_immunity, admin_reject_msg, amv_private_server, default_access

4.1.12. admin_ignore_immunity

`admin_ignore_immunity <#>`

Eure Admins laufen hin und wieder Amok? Ihr habt Ihnen in der users.ini aber leider Immunität verliehen? Das kann man auch allgemein ausschalten, in dem man diesen Wert auf 1 setzt. Die Rache ist Euer, aber Vorsicht, Ihr habt auch keine Immunität mehr! Die Standardeinstellung ist 0 und sollte das auch bleiben.

Beispiel:
`admin_ignore_immunity 1`

Dies setzt die Admin-Immunität außer Kraft.

Siehe auch:
admin_highlander, admin_reject_msg, amv_private_server, default_access

4.1.13. admin_plugin_file

`admin_plugin_file "<string>"`

Der Vorgabewert sollte nur verändert werden, wenn die Datei an einer völlig anderen Stelle als vorgesehen installiert ist. Die meisten Serveradmins sollten die Voreinstellung "addons/adminmod/config/plugin.ini" unbedingt stehen lassen. Steht in admin_plugin_file ein Verzeichnis (z.B. „addons/adminmod/scripts" durchsucht Admin Mod dieses Verzeichnis nach Dateien mit der Endung „.amx" und versucht diese zu laden. Alle nicht benötigten Plugins sind dementsprechend zu löschen oder umzubenennen (z.B. „*.amx.noload").

Beispiel:
`admin_plugin_file "plugin.ini"`

In diesem Fall wird die plugin.ini im Mod-Verzeichnis erwartet. Mehr zu dieser Datei im Kapitel Plugins installieren (plugin.ini).

Siehe auch:
admin_vault_file, ips_file, maps_file, models_file, mysql_dbtable_plugins, pgsql_dbtable_plugins, users_file, words_file

4.1.14. admin_reconnect_timeout

`admin_reconnect_timeout <#>`

Gibt die Zeit (in Sekunden) an, wie lang das Passwort eines Spielers gültig ist, so dass er bei gleicher IP und Namen ohne Neuangabe wieder connecten kann. Den Wert höher als 300 zu setzen, ist ein Sicherheitsrisiko. Ein Wert größer als 0 wird für den Mapwechsel benötigt, ansonsten werden die Admins ihre Rechte verlieren und u.U. vom Server geworfen. Standardeinstellung: 300.

Beispiel:
`admin_reconnect_timeout 60`

Dies setzt die Zeit für die Rückkehr eines Admins auf 60 Sekunden, innerhalb der er sich nicht erneut authentifizieren muss.

Siehe auch:
amv_reconnect_time

4.1.15. admin_reject_msg

`admin_reject_msg "<string>"`

Dieser Text wird zurückgegeben, wenn man ohne Berechtigung versucht Befehle auszuführen.

Beispiel:
`admin_reject_msg "Was wird das? Das darfst Du nicht!"`

Siehe auch:
admin_ignore_immunity, admin_highlander, amv_private_server, default_access, amv_prvt_kick_message, models_kick_msg, nicks_kick_msg, reserve_slots_msg

4.1.16. admin_repeat_freq

`admin_repeat_freq <#>`

Der Wert für admin_repeat_freq gibt in Sekunden an, in welchem Abstand der Text in admin_repeat_msg ausgegeben wird. Bei einem Wert kleiner als 15 startet der Timer beim Mapstart nicht. Eine Änderung des Werts wird erst nach einem Mapchange aktiv wird. (plugin_message) Default: 600.

Beispiel:
`admin_repeat_freq 600`

Die unter admin_repeat_msg hinterlegte Nachricht wird alle 10 Minuten (600 Sekunden) wiederholt.

Siehe auch:
admin_connect_msg, admin_repeat_msg

4.1.17. admin_repeat_msg

`admin_repeat_msg "<string>"`

Gibt den Text an, der als Centersay wiederholt werden soll. Der zeitliche Abstand wird in admin_repeat_freq festgelegt. Setzt man den Wert auf 0, wird keine Nachricht ausgegeben. Solange admin_repeat_freq beim Mapstart nicht kleiner als 15 ist, kann die Nachricht jederzeit wieder aktiviert werden. (plugin_message) Default: „This server is using Admin Mod“

Beispiel:
`admin_repeat_msg "This server is using^nAdmin Mod"`

Dies bringt die Meldung „This server is using Admin Mod“ als Centersay auf den Bildschirm. Allerdings erscheint Admin Mod in der zweiten Zeile, da „^n“ ein Zeilenumbruch ist.

Siehe auch:
admin_connect_msg, admin_repeat_freq

4.1.18. admin_quiet

`admin_quiet <#>`

Hiermit wird festgelegt, ob und wie Admin Mod zurückmelden soll, dass ein Befehl ausgeführt wurde. 0 bedeutet, dass folgende Nachricht erscheint:
`ADMIN Command: <Player> used <command>`.
Bei 1 erscheint nur der Befehl und 2 unterdrückt jegliche Ausgabe. Einige Befehle übergehen aber selbst admin_quiet 2 (z.B. admin_godmode). Voreinstellung ist 0.

Beispiel:
`admin_quiet 2`

Die meisten Serveradmins werden diese Einstellung bevorzugen, bei der nur die Cheat-Möglichkeiten offengelegt werden.

Siehe auch:
admin_connect_msg, admin_repeat_freq

4.1.19. admin_vault_file

`admin_vault_file "<string>"`

Das Vaultfile speichert Einstellungen von Customplugins, sollte also definiert werden. Der Wert "addons/adminmod/config/vault.ini" ist in der Regel nicht zu verändern.

Beispiel:
`admin_vault_file "vault.ini"`

In diesem Fall wird die vault.ini im Mod-Verzeichnis erwartet. Mehr zu dieser Datei im Kapitel Pluginspezifische Einstellungen (vault.ini).

Siehe auch:
admin_plugin_file, ips_file, maps_file, models_file, users_file, words_file

4.1.20. admin_mod_version

`admin_mod_version "<string>"`

Setzt die Admin Mod-Version. Es macht keinen Sinn diesen Wert zu ändern, aber der Vollständigkeit halber soll die Variable hier erwähnt werden. Händische Änderungen werden von Admin Mod überschrieben.

4.1.21. admin_vote_autostart

`admin_vote_autostart <#>`

Für das Plugin hldsld_mapvote wird hier festgelegt, ob 5 Minuten vor Mapchange automatisch ein Mapvote gestartet werden soll. 1 aktiviert die Funktion, 0 deaktiviert sie. Default: 0.

Beispiel:
`admin_vote_autostart 1`

Hiermit ist der Vote vor dem Mapwechsel aktiviert worden.

Siehe auch:
admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, kick_ratio, map_ratio, vote_freq

4.1.22. admin_vote_echo

`admin_vote_echo <#>`

Setzt man die Variable auf 1, wird angezeigt, welcher Spieler welche Option bei einem Vote gewählt hat. Bei 0 wird nur das Endergebnis präsentiert.

Beispiel:
`admin_vote_autostart 1`

Zeigt die Entscheidung jedes einzelnen Spielers beim Vote an.

Siehe auch:
admin_vote_autostart, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, kick_ratio, map_ratio, vote_freq

4.1.23. admin_vote_freq

`admin_vote_freq <#>`

Hiermit wird festgelegt wieviele Sekunden nach einem Mapchange bzw. dem letzten Vote mittels Plugin hldsld_mapvote vergangen sein müssen, bis ein neuer durchgeführt werden darf. Es ist zu beachten, dass dieser Wert nicht für andere Votes gilt. (siehe dazu vote_freq) Standardwert: 600.

Beispiel:
`admin_vote_freq 300`

Diese Einstellung erlaubt einen Mapvote alle 5 Minuten (300 Sekunden).

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, kick_ratio, map_ratio, vote_freq

4.1.24. admin_vote_maxextend

`admin_vote_maxextend <#>`

Legt fest, wie oft eine Map mittels Plugin hldsld_mapvote um 30 Minuten verlängert werden darf. Standardmäßig darf sie das nicht (Einstellung: 0).

Beispiel:
`admin_vote_maxextend 1`

Dies bedeutet, dass die Map einmal um 30 Minuten im Mapvote verlängert werden darf.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_ratio, amv_vote_duration, kick_ratio, map_ratio, vote_freq

4.1.25. admin_vote_ratio

`admin_vote_ratio <#>`

Mit dieser Variablen wird festgelegt, wieviel Prozent der Spieler für einen Mapwechsel stimmen müssen, damit dieser durchgeführt wird (Plugin hldsld_mapvote). Standardwert: 60.

Beispiel:
`admin_vote_ratio 50`

Bei einem Wert von 50 und 10 Spielern auf dem Server müssen 5 für den Mapwechsel stimmen.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, amv_vote_duration, kick_ratio, map_ratio, vote_freq

4.1.26. allow_client_exec

`allow_client_exec <#>`

Plugins können Befehle beim Client ausführen, wenn diese Variable auf 1 gesetzt wird (Standard: 0). Zum Schutz des Clients sind nicht alle Befehle ausführbar. Außerdem wird beim Einloggen eine Nachricht angegeben, dass das Ausführen von Befehlen auf diesem Server erlaubt ist.

Diese Meldung lässt sich weder entfernen noch editieren. Sie dient der Information des Clients, dass theoretisch die Möglichkeit eines böswilligen Eingriffs seitens eines Admins oder Plugins besteht! Leider gibt es immer noch genug "Admins", die ihre Aktionen nicht reflektieren. Dabei bietet Admin Mod doch so viele andere Möglichkeiten der Bestrafung!

Beispiel:
`allow_client_exec 1`

Viele Plugins funktionieren ohne die Aktivierung dieser Funktion nicht. Es empfiehlt sich daher diese Einstellung auf 1 zu setzen.

Siehe auch:
file_access_read, file_access_write

4.1.27. ami_sv_maxplayers

`ami_sv_maxplayers <#>`

Gibt die ursprüngliche Maxplayereinstellung zurück. Diese Variable sollte nicht manuell gesetzt werden, hat nur informativen Charakter für Serverbrowser. Sie dient Serverbrowsern zum erkennen, wieviele Slots für die Admins reserviert sind.

Siehe auch:
amv_hide_reserved_slots, public_slots_free, reserve_slots, reserve_type

4.1.28. amv_anti_cheat_options

`amv_anti_cheat_options "<string>"`

Admin Mod gibt Möglichkeiten an die Hand zwei bestimmte Cheats zu erkennen und Gegenmaßnahmen zu ergreifen.

1. Name Crash Cheat (interessant nur bis HL1108, inzwischen von Valve gefixt)

Der Wert besteht stets aus einer Zeichenfolge (hier nc) und einer Zahl:

“nc0” bedeutet, dass Admin Mod nichts unternimmt (Default)
“nc1” kickt den Spieler vom Server “nc2” kickt den Spieler vom Server und bannt ihn für 24 h

2. Spectator Cheat (vermutlich auch nicht mehr aktuell)

“sp0” bedeutet, dass Admin Mod nichts unternimmt (Default)
“sp1” bannt den Spieler vom Server

Will man beide Optionen gleichzeitig nutzen, so ist als Trennzeichen der Doppelpunkt zu verwenden (z.B. “nc1:sp1”)

Beispiel:
`amv_anti_cheat_options "nc0:sp1"`

Dies aktiviert die Spectator Cheat Erkennung, deaktiviert jedoch die Name Crash Cheat Erkennung.

Siehe auch:
amv_hide_reserved_slots, public_slots_free, reserve_slots, reserve_type

4.1.29. amv_default_config_dir

`amv_default_config_dir "<string>"`

Beinhaltet den Standardpfad der Admin Mod-Installation. Dieser Wert ist zu ändern, falls nicht der Standardpfad benutzt wird (Standard ist “addons/adminmod/config”). Diese Einstellung ist für einige Custom-Plugins relevant.

Beispiel:
`amv_default_config_dir "addons/adminmodtest/config"`

In diesem Beispiel wird erwartet, dass sich die Konfigurationsdateien unüblicherweise im Unterverzeichnis “adminmodtest” befinden.

Siehe auch:
amv_hide_reserved_slots, public_slots_free, reserve_slots, reserve_type

4.1.30. amv_enable_beta

`amv_enable_beta "<string>"`

Einige in Admin Mod eingebaute Funktionen haben nur Betastatus. Daher sind sie zunächst deaktiviert. Einige Plugins setzen solche Funktionen jedoch voraus, oder man selbst will etwas damit experimentieren.

Momentan gibt es drei Betafunktionen. Zur Aktivierung muss eine Zeichenfolge in Kombination mit einer Zahl angegeben werden:

- "menu1" aktiviert die Menüfunktionen unter Admin Mod (wird inzwischen von diversen Plugins benötigt)
- "melog1" Es kommt immer wieder zu verlorenen Entities beim Laden einer Map (fehlende Leiter, fehlende Fenster). Mit melog werden Verdachtsmomente in den Logs hinterlassen.
- "mefix1" Funktioniert wie melog, nur dass statt eines Logeintrages versucht wird, das Problem zu fixen.

Will man mehrere Einträge machen, sind diese mit einem Doppelpunkt zu trennen (z.B. "melog1:mefix1"). Ausschalten kann man die Funktionen durch 0 (z.B. "menu0").

Bitte beachten, dass es sich um Betafunktionen handelt, es also theoretisch zu Problemen kommen kann.

Beispiel:
`amv_enable_beta "menu1:melog0:mefix0"`

Hiermit werden die Menüs aktiviert und die Entity-Funktionen deaktiviert. Dies sollte so auch eingestellt werden. Die Menüs werden bereits vom plugin_CS genutzt, und die Entity-Funktionen haben bislang nicht zum Erfolg geführt.

Siehe auch:
amv_hide_reserved_slots, public_slots_free, reserve_slots, reserve_type

4.1.31. amv_hide_reserved_slots

`amv_hide_reserved_slots <#>`

Wenn aktiviert, werden die reservierten Slots versteckt (1). Bei 0 werden auch diese nach außen angezeigt. Standard: 1.

!!! Es ist zu beachten, dass viele Spieler bei Abschaltung der Funktion irrtümlich davon ausgehen, dass noch Platz auf dem Server ist, obwohl dies nicht der Fall ist. !!! Ein ordentlicher Serverbrowser kann das trotz Aktivierung aber erkennen (Dazu gehört leider nicht der Steam-Browser).

Beispiel:
`amv_hide_reserved_slots 0`

Damit werden auch alle reservierten Slots von außen sichtbar. Notwendig, wenn der Connect nur über den Steam-Browser erfolgt. Nicht zu empfehlen! Besser eine Desktopverknüpfung für den Server erstellen (siehe auch Kapitel zur users.ini).

Siehe auch:
ami_sv_maxplayers, public_slots_free, reserve_slots, reserve_type

4.1.32. amv_log_passwords

`amv_log_passwords <#>`

Ist admin_debug aktiviert und dieser Wert nicht 0, so werden die Admin Mod-Passwörter in die Logs geschrieben. Man kann damit die Passwörter in den Logfiles mitlesen. Dies sollte nur dann aktiviert werden, wenn man beim Connect ein Authentifizierungsproblem hat und überprüfen möchte, ob das Passwort richtig übertragen wird.

Beispiel:
`amv_log_passwords 1`

Dies aktiviert, dass die Passwörter im Klartext in den Logs stehen. Nicht empfehlenswert!!

Siehe auch:
admin_debug

4.1.33. amv_private_server

`amv_private_server <#>`

Hiermit kann man einstellen, dass ausschließlich Spieler, die in der users.ini eingetragen wurden, auf dem Server spielen dürfen. Es ist also vergleichbar mit einem Passwort, nur dass das Passwort nicht weitergegeben werden kann, der Zugang also wirklich exklusiv ist. 1 aktiviert den "privaten" Server. Standard: 0.

Beispiel:
`amv_private_server 1`

Aktiviert, dass der Server nur für Spieler mit einem Eintrag in der users.ini nutzbar ist.

Siehe auch:
admin_ignore_immunity, admin_highlander, admin_reject_msg, default_access, amv_prvt_kick_message

4.1.34. amv_prvt_kick_message

`amv_prvt_kick_message "<string>"`

Legt fest, mit welcher Begründung der Spieler von einem mit amv_private_server festgelegten Server gekickt wird, wenn er nicht in der users.ini steht. Standardmäßig ist kein Text hinterlegt.

Beispiel:
`amv_prvt_kick_message "Server nur fuer Mitglieder! http://reg.xxx.xxx"`

Dem Spieler wird beim Connect mitgeteilt, dass er sich registrieren möge.

Siehe auch:
admin_reject_msg, amv_private_server, models_kick_msg, nicks_kick_msg, reserve_slots_msg

4.1.35. amv_reconnect_time

`amv_reconnect_time <#>`

Wenn ein Admin, beispielsweise beim Mapwechsel, in einen anderen Serverslot rutscht, verliert er die Adminrechte. Mit dieser Variablen kann eingestellt werden, wie lange nach dem Mapwechsel der Admin trotz anderen Slots seine Rechte behält. Während dieser Zeit ist es Admin Mod dann möglich ihn neu zu authentifizieren. Der Maximalwert beträgt 90 Sekunden, bei 0 wird die Funktion abgeschaltet (Standard). Sollte ein Spieler beim Mapwechsel seine Rechte verlieren oder gekickt werden, so sollte versucht werden, den Wert zu erhöhen.

Beispiel:
`amv_reconnect_time 30`

Gibt Admin Mod 30 Sekunden Zeit den Spieler wieder neu zu erkennen.

Siehe auch:
admin_reconnect_timeout

4.1.36. amv_register_cmds

`amv_register_cmds "<string>"`

Plugins für Metamod wie LogD oder Statsme greifen per exec("admin_command ...") auf bestimmte Events zu. Die Verwendung von admin_command ist jedoch auf Grund von Sicherheitsbedenken nicht mehr gestattet. Mit dieser Variablen kann man bestimmten Metamodplugins dennoch Zugriff gewähren. Dazu trägt man hier den entsprechenden String ein, wie "logd_reg". Bei LogD und Statsme ist das allerdings nicht mehr nötig, da diese schon berücksichtigt werden. Bei Verwendung mehrerer Plugins sind die Registrierungsstrings durch Leerzeichen zu trennen.

Beispiel:
`amv_register_cmds "logd_reg sm_reg sm_register"`

Dies erlaubt LogD und Statsme (alt und neue Registrierung) auf admin_command zuzugreifen. Allerdings sind diese schon automatisch berücksichtigt. Der Eintrag wird also nicht benötigt. Andere Metamod-Plugins, die diese Schnittstelle nutzen, sind unbekannt.

4.1.37. amv_vote_duration

`amv_vote_duration <#>`

Gibt die Länge eines Votes in Sekunden an. Normalerweise sind es 30 Sekunden. Minimal sind 2, maximal 1800 Sekunden möglich. Es ist zu beachtetn, dass die Veränderung dieses Wertes keinen Einfluss auf die Länge eines gerade laufenden Votes hat. Erst beim Start des nächsten Votes, werden die Einstellungen aktiv.

Beispiel:
`amv_vote_duration 60`

Damit ist der nächste Vote für 60 Sekunden aktiv.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, kick_ratio, map_ratio, vote_freq

4.1.38. default_access

`default_access <#>`

Mit dieser Variablen wird definiert, welche Rechte die Spieler erhalten, die NICHT in der users.ini stehen. Berechnet wird dieser Wert genauso wie bei einem normalen Benutzer. Übliche Werte sind jedoch entweder 0 oder 1, da darüber hinausgehende Rechte für Nichtadmins keinen Sinn machen. Default: 1.

Beispiel:
`default_access 0`

Die meisten Admins werden den default_access wie gezeigt auf 0 stellen wollen. Einige Rechte aus dem Accesslevel 1 sind meist unerwünscht.

Siehe auch:
admin_ignore_immunity, admin_highlander, admin_reject_msg, amv_private_server

4.1.39. encrypt_password

`encrypt_password <#>`

Admin Mod Passwörter können lokal auf dem Server verschlüsselt werden. Dies verhindert, dass Dritte bei Zugriff auf den Gameserver das Passwort ausspähen können. Es gibt dabei 4 verschiedene Optionen.

- Bei 0 wird nicht verschlüsselt.
- Bei 1 wird das unixeigene Crypt verwendet.
- Bei 2 werden MD5-Hashes verwendet.
- Bei 3 wird die MySQL-Passwort-Funktion verwendet. Diese steht jedoch nur bei Verwendung einer MySQL-Tabelle statt einer users.ini zur Verfügung. Hier gibt es das Problem, dass diese Option nicht mit MySQL 4.1 und neuer funktioniert (außer man setzt den gesamten SQL-Server in den Kompabilitätsmodus). In diesem Fall empfiehlt sich als Alternative, MD5-Hashes zu verwenden.

Zur Erstellung der Passwörter liegen Tools den Admin Mod-Distributionen bei (s. unter encrypt/encrypt.exe).

!!! Bitte beachten, dass der Client stets das Passwort im Klartext angeben muss !!!

Beispiel:
`encrypt_password 2`

In diesem Beispiel wandelt Admin Mod das vom Spieler übertragene Passwort in einen MD5-Hash um und vergleicht diesen mit dem Hash in der users.ini.

Siehe auch:
admin_ignore_immunity, admin_highlander, admin_reject_msg, amv_private_server

4.1.40. file_access_read

`file_access_read <#>`

Mit file_access_read erlaubt man Plugins sämtliche Dateien im Modverzeichnis und seinen Unterverzeichnissen des Servers zu lesen. Viele Plugins brauchen diese Funktion, es sind aber bösartige Plugins nie auszuschließen. Daher ist die Funktion standardmäßig auf 0 gesetzt, aktiviert wird sie durch das Setzen auf 1. Aus diesem Grund ist es immer gut Plugins selber zu compilieren, da der "bösartige" Code schnell auffindbar ist. Grundsätzlich sind uns aber keine solchen Plugins bekannt.

Viele Plugins brauchen aber Zugriff auf einige Dateien. Somit sollte man diese Funktion aktivieren.

Beispiel:
`file_access_read 1`

Aktiviert den Lesezugriff auf Dateien im Modverzeichnis.

Siehe auch:
allow_client_exec, file_access_write

4.1.41. file_access_write

`file_access_write <#>`

Mit file_access_write erlaubt man Plugins sämtliche Dateien im Modverzeichnis und seinen Unterverzeichnissen des Servers zu beschreiben/überschreiben. Viele Plugins brauchen diese Funktion, es sind aber bösartige Plugins nie auszuschließen. Daher ist die Funktion standardmäßig auf 0 gesetzt, aktiviert wird sie durch das Setzen auf 1. Aus diesem Grund ist es immer gut Plugins selber zu compilieren, da der "bösartige" Code schnell auffindbar ist. Grundsätzlich sind uns aber keine solchen Plugins bekannt.

Viele Plugins brauchen aber Zugriff auf einige Dateien. Somit sollte man diese Funktion aktivieren.

Beispiel:
`file_access_write 1`

Aktiviert den Schreibzugriff auf Dateien im Modverzeichnis.

Siehe auch:
allow_client_exec, file_access_read

4.1.42. help_file (veraltet)

`help_file "<string>"`

Wird nur benötigt, wenn man statt das Pluginsystems das alte Scriptfilesystem verwendet. Diese Variable definiert, wo die Datei mit den Hilfebeschreibungen zu finden ist. Das Scriptfile wird schon seit Jahren nicht mehr gepflegt und sollte auch nicht mehr genutzt werden. Der Vollständigkeithalber sollte es aber erwähnt werden.

Siehe auch:
script_file

4.1.43. ips_file

`ips_file "<string>"`

Mittels der Datei ips.ini können festgelegte IPs, IP Ranges oder Subnets Zugriff auf die reservierten Slots erhalten. Bei dieser Variablen wird angegeben, wo sich die Datei ips.ini befindet (z.B. "addons/adminmod/config/ips.ini"). Standardmäßig ist die Funktion abgeschaltet "0".

Beispiel:
`ips_file "addons/adminmod/config/ips.ini"`

Admin Mod sucht mit dieser Einstellung im angegebenen Verzeichnis nach der Datei ips.ini.

Siehe auch:
admin_plugin_file, admin_vault_file, maps_file, models_file, mysql_dbtable_ips, pgsql_dbtable_ips, users_file, words_file

4.1.44. kick_ratio

`kick_ratio <#>`

Legt die Prozentzahl fest, die bei einem Kickvote für den Kick erreicht werden muss, damit dieser durchgeführt wird. Standardeinstellung ist 60.

Beispiel:
`kick_ratio 50`

Stimmen 5 von den 10 auf dem Server befindlichen Spielern für den Kick, so wird derjenige auch gekickt.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, map_ratio, vote_freq

4.1.45. map_ratio

`map_ratio <#>`

Legt die Prozentzahl fest, die bei einem Mapvote für einen Mapwechsel erreicht werden muss, damit dieser durchgeführt wird. Default: 80.

Beispiel:
`map_ratio 40`

Stimmen mindestens 4 von den 10 auf dem Server befindlichen Spielern für den Mapchange, so wird dieser ausgeführt. 80% als Standardwert sind vielleicht etwas hoch gewählt.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, kick_ratio, vote_freq

4.1.46. maps_file

`maps_file "<string>"`

Mit dieser Variablen wird definiert, wo die Datei für die benutzbaren Maps zu finden ist. In der Regel ist dies addons/adminmod/config/maps.ini. Nur die in dieser Datei genannten Maps sind von Admin Mod Plugins nutzbar. Standard: “”

Sollen alle Maps votebar sein, so ist die Datei zu deaktivieren, indem die Variable auf “” gesetzt wird!

Beispiel:
`maps_file "addons/adminmod/config/maps.ini"`

Admin Mod lässt Mapvotes etc. nur zu, wenn die entsprechende Map auch in der maps.ini steht. Ist diese leer, kann gar keine Map ausgewählt werden!

Siehe auch:
admin_plugin_file, admin_vault_file, ips_file, models_file, users_file, words_file

4.1.47. models_file

`models_file "<string>"`

Definiert, wo sich die Datei für die Modelreservierung befindet. Diese Datei wird nur sehr selten verwendet und ist bei Bedarf unter “addons/adminmod/config/models.ini” anzulegen und entsprechend in die Variable zu schreiben. Braucht man keine Modelreservierung sollte diese Variable nicht definiert werden.

Beispiel:
`models_file "addons/adminmod/config/models.ini"`

Der Ort, wo sich die Datei zur Modelreservierung befindet, ist damit definiert.

Siehe auch:
admin_plugin_file, admin_vault_file, ips_file, maps_file, models_kick_msg, mysql_dbtable_models, pgsql_dbtable_models, users_file, words_file

4.1.48. models_kick_msg

`models_kick_msg "<string>"`

Definiert die Kickmessage, die bei Verwendung eines reservierten Models an den Spieler übermittelt wird. (Standard: „[ADMIN] That model is reserved on this server.“)

Beispiel:
`models_kick_msg "So wie Du aussiehst, musst Du draussen bleiben!"`

Dies gibt einen deutlichen Hinweis an den Spieler, wenn er ein reserviertes Model nutzen will.

Siehe auch:
admin_reject_msg, amv_prvt_kick_message, models_file, nicks_kick_msg, reserve_slots_msg

4.1.49. mysql_database (nur MySQL-Version)

`mysql_database "<string>"`

Definiert die Datenbank auf dem MySQL-Server, in der sich die Konfigurationsdaten von Admin Mod befinden.

Beispiel:
`mysql_database "adminmod"`

In diesem Fall sucht Admin Mod die Informationen in der Datenbank „adminmod“.

Siehe auch:
mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.50. mysql_dbtable_ips (nur MySQL-Version)

`mysql_dbtable_ips "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die IPs für die Slotreservierung zu finden sind. Ist kein Admin Mod-MySQL installiert, wird die Variable ignoriert, anderenfalls wird ggf. eine Definition von ips_file übergangen.

Beispiel:
`mysql_dbtable_ips "am_ips"`

Admin Mod erwartet die Daten zur Slotreservierung für IPs in der Tabelle „am_ips“.

Siehe auch:
ips_file, mysql_database, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.51. mysql_dbtable_models (nur MySQL-Version)

`mysql_dbtable_models "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die Modelreservierungen zu finden sind. Ist kein Admin Mod-MySQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von models_file übergangen.

Beispiel:
`mysql_dbtable_models "am_models"`

Admin Mod erwartet die Daten zur Modelreservierung in der Tabelle „am_models“.

Siehe auch:
models_file, mysql_database, mysql_dbtable_ips, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.52. mysql_dbtable_plugins (nur MySQL-Version)

`mysql_dbtable_plugins "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die Pluginpfade zu finden sind. Ist kein Admin Mod-MySQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von admin_plugin_file übergangen.

Beispiel:
`mysql_dbtable_plugins "am_plugins"`

Admin Mod erwartet die Daten zu den Pluginpfaden in der Tabelle „am_plugins“.

Siehe auch:
admin_plugin_file, mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.53. mysql_dbtable_tags (nur MySQL-Version)

`mysql_dbtable_tags "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die reservierten Clantags zu finden sind. Dies ist notwendig, da Regex ein sehr ineffizienter Befehl bei MySQL ist und nicht auf große users-Tabellen angewendet werden sollte.

Beispiel:
`mysql_dbtable_tags "am_tags"`

Admin Mod erwartet die Daten zu den reservierten Clantags in der Tabelle „am_tags“.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.54. mysql_dbtable_users (nur MySQL-Version)

`mysql_dbtable_users "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die Admins, ihre Passwörter und Accesslevels zu finden sind. Ist kein Admin Mod-MySQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von users_file übergangen.

Beispiel:
`mysql_dbtable_users "am_users"`

Admin Mod erwartet die Daten zu den Admins in der Tabelle „am_users".

Siehe auch:
users_file, mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.55. mysql_dbtable_words (nur MySQL-Version)

`mysql_dbtable_words "<string>"`

Diese Variable legt fest, in welcher MySQL-Tabelle die zu zensierenden Wörter zu finden sind. Ist kein Admin Mod-MySQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von words_file übergangen.

Beispiel:
`mysql_dbtable_words "am_words"`

Admin Mod erwartet die Daten zu den Admins in der Tabelle „am_words".

Siehe auch:
words_file, mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.56. mysql_host (nur MySQL-Version)

`mysql_host "<string>"`

Hiermit wird festgelegt, auf welchem Rechner Admin Mod-MySQL die MySQL-Datenbank findet (z.B. „localhost“). Wird MySQL nicht verwendet, wird diese Variable ignoriert.

Beispiel:
`mysql_host "localhost"`

In diesem Fall erwartet Admin Mod auf dem gleichen Rechner die MySQL-Datenbank.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.57. mysql_pass (nur MySQL-Version)

`mysql_pass "<string>"`

Mit dieser Variablen wird das Passwort für den Zugang zum MySQL-Datenbank festgelegt. Standard: „“.

Beispiel:
`mysql_pass "geheim"`

Das Passwort für den Zugriff auf die MySQL-Datenbank ist auf „geheim“ gesetzt. Bitte ein besseres Passwort benutzen.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_preload, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.58. mysql_preload (nur MySQL-Version)

`mysql_preload <#>`

Mit dieser Variable wird eingestellt, ob die Datenbank-Tabellen beim Admin Mod-Start in den Speicher geladen werden sollen. Dies verringert ganz erheblich die Last auf dem MySQL-Server, da keine Querys mehr ausgeführt werden müssen, führt aber beispielsweise bei mehr als 1000 Usern in der users-Tabelle zu einer übermäßigen Nutzung von Speicher durch Admin Mod. 1 schaltet den preload ein. Außerdem ist die Einstellung 0 notwendig um die Variablen mysql_tags_sql und mysql_users_sql zu nutzen.

Beispiel:
`mysql_preload 1`

Diese Einstellung aktiviert das Laden der User in den Speicher beim Serverstart und beim Mapwechsel (wie users.ini). Sie deaktiviert mysql_tags_sql und mysql_users_sql!

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_tags_sql, mysql_user, mysql_users_sql

4.1.59. mysql_tags_sql (nur MySQL-Version)

`mysql_tags_sql "<string>"`

Admin Mod lässt es dem Benutzer frei, den SQL-Befehl beim Zugriff auf die tags-Tabelle der MySQL-Datenbank nach eigenem Wunsch zu verändern (z.B. zur Implementierung in eine Forumssoftware).

Standard:
`"SELECT pass,access FROM %s WHERE '%s' REGEXP nick OR nick = '%s'"`

Beispiel:
`mysql_tags_sql "SELECT password, access FROM %s WHERE tag REGEXP '%s' OR tag = '%s'"`

Der Ausdruck muss stets drei "%s" enthalten. Diese werden in der Reihenfolge Tags-Table (mysql_dbtable_tags), Username und STEAM_ID gefüllt. mysql_preload muss auf 0 gesetzt sein, um diese Variable zu nutzen.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_user, mysql_users_sql

4.1.60. mysql_user (nur MySQL-Version)

`mysql_user "<string>"`

Definiert den Loginnamen für die MySQL-Datenbank.

Beispiel:
`mysql_user "adminmodadmin"`

Der Username für den Zugriff auf die MySQL-Datenbank ist auf „adminmodadmin“ gesetzt.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_users_sql

4.1.61. mysql_users_sql (nur MySQL-Version)

`mysql_users_sql "<string>"`

Admin Mod lässt es dem Benutzer frei, den SQL-Befehl beim Zugriff auf die users-Tabelle der MySQL-Datenbank nach eigenem Wunsch zu verändern (z.B. zur Implementierung in eine Forumssoftware).

Standard:
`"SELECT pass,access FROM %s where nick='%s' or nick='%s'"`

Beispiel:
```
mysql_users_sql "SELECT password, access FROM %s WHERE username = '%s'
OR username = '%s'"
```

Der Ausdruck muss stets drei „%s“ enthalten. Diese werden in der Reihenfolge User-Table (mysql_dbtable_users), Username und STEAM_ID gefüllt. mysql_preload muss auf 0 gesetzt sein, um diese Variable zu nutzen.

Siehe auch:
mysql_database, mysql_dbtable_ips, mysql_dbtable_models, mysql_dbtable_plugins, mysql_dbtable_tags, mysql_dbtable_users, mysql_dbtable_words, mysql_host, mysql_pass, mysql_preload, mysql_tags_sql, mysql_user

4.1.62. nicks_kick_msg

`nicks_kick_msg "<string>"`

Man legt hier fest, welche Meldung der Client bekommt, wenn er gegen die Namensreservierung verstößt.

Standard: `"[ADMIN] That name is reserved on this server."`

Beispiel:
`nicks_kick_msg "Du kommst hier ned rein! Nickname!"`

Der Spieler wird darauf hingewiesen, dass er auf Grund seines Namens gekickt wurde.

Siehe auch:
admin_reject_msg, amv_prvt_kick_message, models_kick_msg, reserve_slots_msg, users_file

4.1.63. password_field

`password_field "<string>"`

Ein wichtige Variable, damit sich die Spieler als Admin anmelden können. Hiermit legt man fest, unter welchem Variablennamen Admin Mod vom Spieler das Passwort erwartet. Standard: "_pw-home"

Beispiel:
`password_field "_ichwillrein"`

Dann lautet die setinfo-Zeile beim Client in etwa so:

`setinfo "_ichwillrein" "binichschondrin"`

!!! Es ist zu beachten, dass das Passwortfeld aus Sicherheitsgründen stets mit einem Unterstrich beginnt. Anderenfalls wird Admin Mod den Server nicht starten lassen !!!

Siehe auch:
mysql_dbtable_users, users_file

4.1.64. pgsql_database (nur PostgreSQL-Version)

`pgsql_database "<string>"`

Definiert die Datenbank auf dem PostgreSQL-Server, in der sich die Konfigurationsdaten von Admin Mod befinden.

Beispiel:
`pgsql_database "adminmod"`

In diesem Fall sucht Admin Mod die Informationen in der Datenbank „adminmod".

Siehe auch:
pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.65. pgsql_dbtable_ips (nur PostgreSQL-Version)

`pgsql_dbtable_ips "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die IPs für die Slotreservierung zu finden sind. Ist kein Admin Mod-PostgreSQL installiert, wird die Variable ignoriert, anderenfalls wird ggf. eine Definition von ips_file übergangen.

Beispiel:
`pgsql_dbtable_ips "am_ips"`

Admin Mod erwartet die Daten zur Slotreservierung für IPs in der Tabelle „am_ips".

Siehe auch:
ips_file, pgsql_database, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.66. pgsql_dbtable_models (nur PostgreSQL-Version)

`pgsql_dbtable_models "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die Modelreservierungen zu finden sind. Ist kein Admin Mod-PostgreSQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von models_file übergangen.

Beispiel:
`pgsql_dbtable_models "am_models"`

Admin Mod erwartet die Daten zur Modelreservierung in der Tabelle „am_models“.

Siehe auch:
models_file, pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.67. pgsql_dbtable_plugins (nur PostgreSQL-Version)

`pgsql_dbtable_plugins "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die Pluginpfade zu finden sind. Ist kein Admin Mod-PostgreSQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von admin_plugin_file übergangen.

Beispiel:
`pgsql_dbtable_plugins "am_plugins"`

Admin Mod erwartet die Daten zu den Pluginpfaden in der Tabelle „am_plugins“.

Siehe auch:
admin_plugin_file, pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.68. pgsql_dbtable_tags (nur PostgreSQL-Version)

`pgsql_dbtable_tags "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die reservierten Clantags zu finden sind.

Beispiel:
`pgsql_dbtable_tags "am_tags"`

Admin Mod erwartet die Daten zu den reservierten Clantags in der Tabelle „am_tags“.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.69. pgsql_dbtable_users (nur PostgreSQL-Version)

`pgsql_dbtable_users "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die Admins, ihre Passwörter und Accesslevels zu finden sind. Ist kein Admin Mod-PostgreSQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von users_file übergangen.

Beispiel:
`pgsql_dbtable_users "am_users"`

Admin Mod erwartet die Daten zu den Admins in der Tabelle „am_users“.

Siehe auch:
users_file, pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.70. pgsql_dbtable_words (nur PostgreSQL-Version)

`pgsql_dbtable_words "<string>"`

Diese Variable legt fest, in welcher PostgreSQL-Tabelle die zu zensierenden Wörter zu finden sind. Ist kein Admin Mod-PostgreSQL installiert, wird die Variable ignoriert, anderenfalls wird eine Definition von words_file übergangen.

Beispiel:
`pgsql_dbtable_words "am_words"`

Admin Mod erwartet die Daten zu den Admins in der Tabelle „am_words“.

Siehe auch:
words_file, pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.71. pgsql_host (nur PostgreSQL-Version)

`pgsql_host "<string>"`

Hiermit wird festgelegt, auf welchem Rechner Admin Mod-PostgreSQL die PostgreSQL-Datenbank findet (z.B. „localhost“). Wird PostgreSQL nicht verwendet, wird diese Variable ignoriert.

Beispiel:
`pgsql_host "localhost"`

Hier erwartet Admin Mod auf dem gleichen Rechner die PostgreSQL-Datenbank.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.72. pgsql_pass (nur PostgreSQL-Version)

`pgsql_pass "<string>"`

Mit dieser Variablen wird das Passwort für den Zugang zum PostgreSQL-Datenbank festgelegt. Standard: „“.

Beispiel:
`pgsql_pass "geheim"`

Das Passwort für den Zugriff auf die PostgreSQL-Datenbank ist auf „geheim“ gesetzt. Bitte ein besseres Passwort benutzen.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.73. pgsql_port (nur PostgreSQL-Version)

`pgsql_port "<string>"`

Hiermit setzt man den Port mit dem auf die PostgreSQL-Datenbank zugegriffen werden soll. Default: 5432

Beispiel:
`pgsql_port 5433`

Dies verändert den Standardport für den Zugang zur PostgreSQL-Datenbank auf 5433.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_preload, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.74. pgsql_preload (nur PostgreSQL-Version)

`pgsql_preload <#>`

Mit dieser Variable wird eingestellt, ob die Datenbank-Tabellen beim Admin Mod-Start in den Speicher geladen werden sollen. Dies verringert ganz erheblich die Last auf dem PostgreSQL-Server, da keine Querys mehr ausgeführt werden müssen, führt aber beispielsweise bei mehr als 1000 Usern in der users-Tabelle zu einer übermäßigen Nutzung von Speicher durch Admin Mod. 1 schaltet den preload ein. Außerdem ist die Einstellung 0 notwendig um die Variablen pgsql_tags_sql und pgsql_users_sql zu nutzen.

Beispiel:
`pgsql_preload 1`

Diese Einstellung aktiviert das Laden der User in den Speicher beim Serverstart und beim Mapwechsel (wie users.ini). Sie deaktiviert pgsql_tags_sql und pgsql_users_sql!

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_tags_sql, pgsql_user, pgsql_users_sql

4.1.75. pgsql_tags_sql (nur PostgreSQL-Version)

`pgsql_tags_sql "<string>"`

Admin Mod lässt es dem Benutzer frei, den SQL-Befehl beim Zugriff auf die tags-Tabelle der PostgreSQL-Datenbank nach eigenem Wunsch zu verändern (z.B. zur Implementierung in eine Forumssoftware).

Standard:
`"SELECT pass,access FROM %s WHERE '%s' REGEXP nick OR nick = '%s'"`

Beispiel:
```
pgsql_tags_sql "SELECT password, access FROM %s WHERE tag REGEXP '%s' OR
tag = '%s'"
```

Der Ausdruck muss stets drei "%s" enthalten. Diese werden in der Reihenfolge Tags-Table (pgsql_dbtable_tags), Username und STEAM_ID gefüllt. pgsql_preload muss auf 0 gesetzt sein, um diese Variable zu nutzen.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_user, pgsql_users_sql

4.1.76. pgsql_user (nur PostgreSQL-Version)

`pgsql_user "<string>"`

Definiert den Loginnamen für die PostgreSQL-Datenbank.

Beispiel:
`pgsql_user "adminmodadmin"`

Der Username für den Zugriff auf die PostgreSQL-Datenbank ist auf „adminmodadmin“ gesetzt.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_users_sql

4.1.77. pgsql_users_sql (nur PostgreSQL-Version)

`pgsql_users_sql "<string>"`

Admin Mod lässt es dem Benutzer frei, den SQL-Befehl beim Zugriff auf die users-Tabelle der PostgreSQL-Datenbank nach eigenem Wunsch zu verändern (z.B. zur Implementierung in eine Forumssoftware).

Standard:
`"SELECT pass,access FROM %s where nick='%s' or nick='%s' "`

Beispiel:
```
pgsql_users_sql "SELECT password, access FROM %s WHERE username = '%s'
OR username = '%s' "
```

Der Ausdruck muss stets drei „%s“ enthalten. Diese werden in der Reihenfolge User-Table (pgsql_dbtable_users), Username und STEAM_ID gefüllt. pgsql_preload muss auf 0 gesetzt sein, um diese Variable zu nutzen.

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user

4.1.78. pretty_say

`pretty_say <#>`

Pretty_say definiert, wie ein Centersay dargestellt werden soll. Steht dies auf 1 (Standard), so wird der Centersay farbig und mit Fade-In und -Out dargestellt, bei 0 als simpler Text. Standard: 1

Beispiel:
`pretty_say 1`

Diese Einstellung bringt Farbe in Admin Mod (aber dezent...).

Siehe auch:
pgsql_database, pgsql_dbtable_ips, pgsql_dbtable_models, pgsql_dbtable_plugins, pgsql_dbtable_tags, pgsql_dbtable_users, pgsql_dbtable_words, pgsql_host, pgsql_pass, pgsql_port, pgsql_preload, pgsql_tags_sql, pgsql_user

4.1.79. public_slots_free

`public_slots_free <#>`

Diese Variable zeigt die noch freien Plätze auf dem Server an. Ein manuelles Setzen ist nutzlos und hat außer einer Fehldarstellung keine Auswirkungen. Rein informative Variable für Serverbrowser.

Siehe auch:
ami_sv_maxplayers, amv_hide_reserved_slots, reserve_slots, reserve_type

4.1.80. reserve_slots

`reserve_slots <#>`

Man legt hier fest, wieviele Plätze (Slots) für Administratoren reserviert werden. Bitte auch reserve_type beachten. Reservierte Slots können nur von Admins mit dem Recht 32768 genutzt werden. Die reservierten Slots sind standardmäßig abgeschaltet (Default: 0).

Beispiel:
`reserve_slots 1`

Diese Einstellung macht einen Serverslot nur für die Admins nutzbar. Mehr zu dem Thema ist dem Kapitel 4.4.4 (Serverplätze reservieren) zu entnehmen.

Siehe auch:
ami_sv_maxplayers, amv_hide_reserved_slots, public_slots_free, reserve_slots_msg, reserve_type

4.1.81. reserve_slots_msg

`reserve_slots_msg "<string>"`

Legt fest, welche Nachricht angezeigt wird, wenn nur noch reservierte Slots vorhanden sind und ein normaler Spieler versucht auf den Server zu connecten. Standardmäßig bekommt der Spieler folgendes zu lesen:

`"There are no reserved slots available on the server."`

Beispiel:
`reserve_slots_msg "Alle freien Plaetze sind schon besetzt. Sorry!"`

Ein freundlicher Hinweis, dass leider keine freien Plätze mehr frei sind. Mehr zu dem Thema ist dem Kapitel 4.4.4 (Serverplätze reservieren) zu entnehmen.

Siehe auch:
reserve_slots, reserve_type

4.1.82. reserve_type

`reserve_type <#>`

Diese Variable ist zunächst komplexer als sie erscheint. Sie legt fest, wie Admin Mod mit den reservierten Slots umgehen soll.

- 0: Admins werden zunächst in öffentlichen Slots untergebracht und die reservierten Slots werden bei einem Disconnect zuerst freigemacht.
- 1: Unabhängig vom Wert in reserve_slots ist immer 1 Slot zum Connecten reserviert.
- 2: Connectende Admins werden zunächst in die reservierten Slots geschoben und bei Disconnects zunächst öffentliche Slots geleert.

Beispiel:
`reserve_type 1`

Einen Slot ausschließlich zum Connecten der Admins zu verwenden, ist von den meisten gewünscht. Bei dieser Einstellung wird die Einstellung von reserve_slots ignoriert und nur ein Slot reserviert. Der Platz bleibt immer frei, steht also nicht zum Spielen zur Verfügung. Er ist nur temporär für die Zeitdauer des Connectens für die Admins nutzbar. Für den Admin wird der Spieler mit dem höchsten Ping gekickt. Mehr zu dem Thema ist dem Kapitel 4.4.4 (Serverplätze reservieren) zu entnehmen.

Siehe auch:
ami_sv_maxplayers, amv_hide_reserved_slots, public_slots_free, reserve_slots, reserve_slots_msg

4.1.83. script_file (veraltet)

`script_file "<string>"`

Definiert, wo das Scriptfile zu finden ist. Die Verwendung des Scriptfiles ist veraltet, es wird stattdessen das Pluginsystem (admin_plugin_file) verwendet.

Siehe auch:
admin_plugin_file, help_file

4.1.84. use_regex

`use_regex <#>`

Schaltet RegEx (Regular Expressions; reguläre Ausdrücke) für die Verwendung in der Datei users.ini ein (1) bzw. aus (0). RegEx ermöglicht die Verwendung von Wildcards und kann z.B. für die Reservierung von Clantags genutzt werden. Regex ist ein mächtiges aber auch kompliziertes Tool, was meist eine Überarbeitung der gesamten users.ini mit sich zieht. Es lohnt sich aber. Standardeinstellung: 0.

Beispiel:
`use_regex 1`

Schaltet die Verwendung von RegEx ein.

Das Kapitel 4.4.6 (RegEx) geht genauer auf das Thema ein.

Siehe auch:
admin_plugin_file, help_file

4.1.85. users_file

`users_file "<string>"`

Hier wird festgelegt, wo sich die Datei mit den Adminrechten befindet (in der Regel „addons/adminmod/config/users.ini“).

Beispiel:
`users_file "users.ini"`

Mit dieser Einstellung erwartet Admin Mod die users.ini im Mod-Verzeichnis.

Siehe auch:
admin_plugin_file, admin_vault_file, ips_file, maps_file, models_file, mysql_dbtable_users, nicks_kick_msg, pgsql_dbtable_users, words_file

4.1.86. vote_freq

`vote_freq <#>`

Mit dieser Variable wird festgelegt, welcher Zeitraum (Sekunden) zwischen zwei Votes bzw. nach einem Mapchange verstrichen sein müssen, damit ein neuer Vote erstellt werden kann. Setzt man den Wert auf 0 wird Voten komplett abgeschaltet. Es ist zu beachten, dass der Wert beim Ändern erst nach dem nächsten Vote aktiv wird. Standard: 180.

Beim Plugin hldsld_mapvote ist eine andere Variable zu setzen (admin_vote_freq).

Beispiel:
`vote_freq 300`

Stellt die Zeit, die zwischen zwei Votes zu verstreichen hat, auf 5 Minuten.

Siehe auch:
admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration, kick_ratio, map_ratio, admin_vote_kick, admin_vote_map, admin_vsay

4.1.87. words_file

`words_file "<string>"`

Die Variable definiert den Ort, an dem die Datei mit den zu zensierenden Wörtern zu finden ist (in der Regel „addons/adminmod/config/wordlist.txt“).

Beispiel:
`words_file "wordlist.txt"`

Mit dieser Einstellung erwartet Admin Mod die wordlist.txt im Mod-Verzeichnis.

Siehe auch:
admin_plugin_file, admin_vault_file, ips_file, maps_file, models_file, mysql_dbtable_users, nicks_kick_msg, pgsql_dbtable_users, users_file

4.2. Plugins installieren (plugin.ini)

Einer der Gründe für die Beliebtheit von Adminmod ist sicherlich das Pluginsystem. Einige Plugins, die die Standardfunktionen beinhalten, werden bereits mit der Admin Mod Distribution geliefert. Standardmäßig erwartet Admin Mod die Plugins im Verzeichnis „addons/adminmod/scripts". Admin Mod lädt aber nicht alle Dateien aus diesem Verzeichnis. In der „plugin.ini" (Bitte nicht mit der plugins.ini von Metamod verwechseln!) werden die Dateien festgelegt, die geladen werden sollen, und wo sie relativ zum Modverzeichnis zu finden sind.

Standardmäßig sollte nach einer normalen Installation die plugin.ini folgendermaßen aussehen:

```
addons/adminmod/scripts/plugin_antiflood.amx
addons/adminmod/scripts/plugin_base.amx
addons/adminmod/scripts/plugin_chat.amx
addons/adminmod/scripts/plugin_cheat.amx
# addons/adminmod/scripts/plugin_CS.amx
# addons/adminmod/scripts/plugin_TFC.amx
addons/adminmod/scripts/plugin_hldsld_mapvote.amx
addons/adminmod/scripts/plugin_message.amx
addons/adminmod/scripts/plugin_retribution.amx
addons/adminmod/scripts/plugin_fun.amx
```

Hier liegt folgendes Prinzip vor:
„addons/adminmod/scripts" ist der Ordner, in dem in der Regel die Plugins liegen. Aber es ist auch jedes andere Verzeichnis relativ zum Modverzeichnis möglich (auch außerhalb mit „../"). Es schließt sich der Name des Plugins an (Wie immer gilt unter Linux die Groß- und Kleinschreibung zu beachten!). Soll ein bestimmtes Plugin nicht geladen werden, ist eine Raute (#) der Zeile voranzustellen und somit auszukommentieren (s. plugin_CS). Alternativ kann die entsprechende Zeile auch einfach gelöscht werden.

WICHTIG: Um Änderungen an der plugin.ini dem Server bekannt zu geben, ist ein Serverrestart oder besser ein einfacher Mapwechsel notwendig. Erst dann wird die Datei neu eingelesen.

Mittels der Include-Anweisung können aber auch weitere Dateien eingebunden werden. Es ist dabei sogar möglich diese Dateien außerhalb des Mod-Verzeichnisses einzubinden.

```
#include "../../../../standard-plugin.ini"
```

In der beschriebenen „standard-plugin.ini" (hier im hlds-Verzeichnis) könnten beispielsweise Plugins stehen, die auf allen Gameservern laufen sollen. Ein identischer Eintrag bei allen Gameservern erlaubt die zentrale Verwaltung der Plugins über diese Datei.

hlds/cstrike/addons/adminmod/config/plugin.ini:

```
#include "../../../../standard-plugin.ini"
addons/adminmod/scripts/plugin_CS.amx
```

hlds/standard-plugin.ini:
```
../scripts/plugin_antiflood.amx
../scripts/plugin_base.amx
../scripts/plugin_chat.amx
../scripts/plugin_cheat.amx
../scripts/plugin_hldsld_mapvote.amx
../scripts/plugin_message.amx
../scripts/plugin_retribution.amx
../scripts/plugin_fun.amx
```

Die Plugins aus der „standard-plugin.ini“ befinden sich dann in „hlds/scripts“. Wie man erkennen kann, können Includes auch mit normalen Einträgen kombiniert werden. In diesem Fall könnte man annehmen, dass ein CS, TFC und ein Ricochet-Server läuft. Je nach Mod kann man dann die Spezialplugins aus dem normalen Scripts-Verzeichnis starten während die übergreifenden aus dem gemeinsamen Verzeichnis geladen werden.

Hier auch noch der Hinweis. Für Counter-Strike und Team Fortress Classic gibt es spezielle Standardplugins, die zunächst erst aktiviert werden müssen (Entfernen der Raute „#“). Gerade bei Counter-Strike kommen oftmals Fragen, warum denn die Befehle admin_restartround oder admin_ct nicht funktionieren. Diese Befehle gibt es erst, wenn man plugin_CS auch aktiviert.

Es gibt darüber hinaus auch die Möglichkeit alle Dateien aus einem entsprechenden Verzeichnis zu laden. Die ungewünschten Plugins müssen dann natürlich entfernt oder umbenannt werden (z.B. „*.amx“ in „*.amx.noload“). Des Weiteren muss die Variable admin_plugin_file das Verzeichnis der Scripts und nicht die plugin.ini beinhalten. Damit kann die plugin.ini entfallen, man kann dann aber die Ladereihenfolge nur noch über den Dateinamen beeinflussen, was manchmal unerwünscht ist.

4.3. Plugins kompilieren

Es taucht immer wieder die Frage auf, wie man aus einer sma- eine amx-Datei macht. Man nennt das Kompilieren. Dabei ist „.sma“ der Quellcode und „.amx“ das fertige Plugin.

Jeder sollte grundsätzlich seine Plugins selber compilieren. Das ist nicht wirklich schwierig und verringert das Risiko, dass einem ein bösartiges Plugin untergeschoben wird.

Man braucht zunächst einen Compiler. Dieser liegt der jeweils aktuellen Admin Mod-Version bei.

Nach dem Entpacken wechselt man in das Verzeichnis „scripting/myscripts“. Dies ist das Verzeichnis für die Quellcode-Dateien (sma). Zwei Dateien sind bereits vorhanden. Unter Linux sind das „compile“ und „compile_all“ sowie unter Windows „compile.bat“ und „compile_all.bat“.

Um einzelne Plugins zu compilieren muss man eine Shell bzw. Eingabeaufforderung öffnen, und im myscripts-Verzeichnis schreiben:

```
./compile plugin_xxx.sma
```

bzw. für Windows

```
compile.bat plugin_xxx.sma
```

Wobei plugin_xxx.sma natürlich für jedes x-beliebige Plugin steht.

Das ist etwas unbequem. Weil der Compiliervorgang meist recht kurz ist, kann man auch alle Plugins auf einmal kompilieren:

```
./compile_all
```

bzw. für Windows

```
compile_all.bat
```

Sofern keine Fehlermeldungen (ERROR) aufgetreten sind, muss man nun nur noch ins Verzeichnis „scripting/mybinaries“ gehen und die fertigen Plugins abholen.

Der Unterschied zwischen ERROR und WARNING ist zu beachten. Plugins mit WARNINGs funktionieren meist ohne Probleme. Es ist zwar unschön, wenn man diese bekommt, aber es ist kein Weltuntergang. In der Regel passiert das gern, wenn man ältere Plugins compilieren möchte. Der Compiler wurde von Version zu Version verschärft. Da sind Warnings dann nicht auszuschließen.

Es darf weiterhin nicht übersehen werden, dass amx-Dateien nicht gleich amx-Dateien sind. Sie wurden für das jeweilige Betriebssystem kompiliert, auf dem der Compiler lief.

Viele kompilieren unter Windows, haben aber einen Linuxserver. Seit Version 2.50.56 konvertiert Admin Mod zwar die Plugins selbstständig bei jedem Mapwechsel Plugins. Das kostet aber wiederum bei jedem Mapwechsel ein wenig Rechenzeit.

Als verantwortungsbewusster Administrator sollte man daher die Plugins vorab konvertieren. Die entsprechenden Konverter gibt es auf adminmod.org[1] und adminmod.de[2].

Ausgeführt werden die Kommandozeilen-Versionen (CMD) folgendermaßen:

```
./amxconvert plugin_xxx.amx
```

bzw. für Windows

```
amxconvert.exe plugin_xxx.amx
```

Für Windows gibt es aber auch eine Version mit einer graphischen Oberfläche (GUI).

[1] http://www.adminmod.org/index.php?go=downloads#tools
[2] http://forum.adminmod.de/info.php?go=tools.html

4.4. Administratoren einrichten (users.ini)

4.4.1. Serverseitige Einstellungen

Administratoren müssen über die users.ini eingerichtet werden, denn zunächst weiß der Server nicht, wer Admin ist. Außerdem müssen auch die Rechte festgelegt werden. Die users.ini ist in „addons/adminmod/config“ zu finden. In die users.ini muss entweder ein Spielername, eine ID oder eine IP, sowie das dazugehörige Passwort und die Rechte des Admins eingetragen werden.

Typische Einträge in der users.ini sehen in etwa so aus:

```
[Bond]JamesBond:007:131071
192.168.14.31:IloveNY:255
STEAM_0:0:12345:steamy:134
\:Mr. Colon\::colonizeme:1492
```

Das Trennzeichen für die Einträge ist der Doppelpunkt.

4.4.1.1. *Player*:Password:Rights

Der Eintrag vor dem ersten Doppelpunkt kann, wie den Beispielen oben zu entnehmen ist,

einen Spielernamen,
eine WONID,
eine IP-Adresse,
eine STEAM_ID oder
eine VALVE_ID

aufnehmen. Eine Kombination von unterschiedlichen IDs oder Spielernamen in einer users.ini ist ohne Probleme möglich. Kommen Doppelpunkte im Spielernamen auf müssen diese mit einem Backslash „\“ escaped werden (s. viertes Beispiel). Die Doppelpunkte in den Steam IDs werden automatisch erkannt und müssen nicht escaped werden. Wird die Option hyperref[use-regex]use_regex genutzt, müssen noch weitere Zeichen escaped werden. Darauf wird aber im Kapitel 4.4.6 (RegEx (Clantags reservieren etc.)) näher eingegangen.

Von der Verwendung der Spielernamen zur Authentifizierung bei Admin Mod sollte Abstand genommen werden. Jeder kann einen in der users.ini angelegten Namen annehmen und mit dem richtigen Passwort hat er bereits Rechte auf dem Server. Außerdem verliert man bei einem Namenswechsel die Adminrechte.

Weiterhin sollte die Verwendung von IPs auf Netze mit statischer IP-Vergabe beschränkt werden, also meist LANs mit statischen IPs (keine automatische Vergabe) oder auf MAC-Adresse basierte Vergabe von IPs mittels DHCP.

Seit der Admin Mod Version 2.50.60 ist die localhost Adresse, 127.0.0.1, als ID für den Eröffner eines Listenservers erlaubt. Da zwar jeder diese Adresse auf seinem Rechner erhält, sie aber nur auf dem eigenen gültig ist, ist dies eine sichere Methode, so dass ein Passwort ausgelassen werden kann. Es ist deshalb auch nicht notwendig eine adminpass.cfg anzulegen. Der Einloggvorgang erfolgt automatisch.

Der users.ini Eintrag lautet dann:

```
127.0.0.1::131071
```

Dem Eröffner sollten natürlich sämtliche Rechte, 131071, gegeben werden. Letztlich könnte er auch admin_cmd vor die Admin Mod Befehle setzen. Eine Einschränkung wäre daher unnütz.

4.4.1.2. Player:*Password*:Rights

Der mittlere Eintrag beinhaltet das Passwort des Spielers.

Das Passwort in der users.ini kann aber auch leergelassen werden, so dass beim Client keine Einstellungen vorgenommen werden müssen. Diese Methode sollte nur bei absoluten DAUs (Dümmste Anzunehmende User) verwendet werden, die mit den Clienteinstellungen nicht zurecht kommen. Aber eigentlich sollten solche Leute schon deshalb keine Adminrechte bekommen. Man kann aber zumindest bei der Verwendung der Steam ID recht sicher sein, dass niemand sie übernehmen kann. 100%ig sicher sollte man sich aber nicht sein. Bei allen anderen Spielereinträgen wie IP oder Name ist das Weglassen des Passwortes in jedem Fall fahrlässig!

Beispiel für das Weglassen des Passworts:

```
STEAM_0:1:9174::131071
```

Umgekehrt kann man die Sicherheit auch erhöhen, indem man verschlüsselte Passwörter verwendet.

Dies ist besonders anzuraten, wenn man seinen Gameserver auf einem Rechner hat, der noch von anderen Personen genutzt wird. Es wird verhindert, dass diese die Admin Mod-Passwörter ausspähen können.

Bis Admin Mod 2.50.56 war dies nur mit Linuxrechnern möglich. Ein kleines Tool namens make_pass, das der Linuxdistribution beiliegt, wird zur Erstellung der verschlüsselten Passwörter verwendet. Der Rückgabewert von `./make_pass meinpasswort` ist dann die Verschlüsselung für „meinpasswort“. Das verschlüsselte Passwort muss nun statt des Klartext-Passworts in die users.ini aufgenommen und die Variable encrypt_password 1 in der adminmod.cfg gesetzt werden.

Mit der Version 2.50.56 wurde dieses Tool durch das Programm encrypt ersetzt. Dieses ist nun in der Lage die alten Passwörter (Parameter -c) bzw. auch MD5-Hashes (Parameter -m) zu produzieren. Die encryt_password Variable ist abwärtskompatibel, muss jedoch auf 2 gesetzt werden, wenn man MD5-Hashes nutzen möchte.

Inzwischen gibt es das Verschlüsselungstool auch mit einer graphischen Oberfläche (sowohl unter Linux als auch unter Windows), die auf adminmod.de[3] zum Download zur Verfügung stehen.

Beim Client ist nichts zu ändern. Er überträgt das Passwort weiterhin im Klartext. Wenn jemand den Netzwerkverkehr mitliest, kann er das Passwort immer noch das Passwort ermitteln. Man sollte also ein Passwort wählen, das man ansonsten nicht verwendet. Die Verschlüsselung dient ausschließlich dem Entgegenwirken eines Ausspähens lokal auf dem Server.

Wichtig: Es ist nicht möglich verschiedene Verschlüsselungsarten oder eine Verschlüsselung in Kombination mit Klartextpasswörter gleichzeitig in der users.ini zu verwenden. Man muss ich für eine Variante entscheiden.

4.4.1.3. Player:Password:*Rights*

Der letzte Eintrag definiert die Rechte des Admins. Diese können ganz unterschiedlicher Art sein. Nicht jeder Admin benötigt schließlich volle Zugangsrechte zum Server.

Adminmod hat daher mehrere Rechtelevel. Jeder einzelne Level beinhaltet Befehle, die der Admin dann ausführen darf. Die Level werden durch Zahlen symbolisiert. Ein Rechtelevel der Stufe 2 beinhaltet z.B. die Befehle: admin_map und admin_timelimit. Ein Admin, der diesen Rechtelevel besitzt, darf also z.B. die Map wechseln oder das Zeitlimit ändern. Die Rechtelevel werden ADDIERT und die Summe in die users.ini eingetragen. Mehr dazu im Kapitel 4.4.3 (Rechtelevel).

WICHTIG: Änderungen an der users.ini werden erst nach einem Serverrestart, Mapwechsel oder dem Befehl "admin_reload" übernommen.

Weitere Dateien lassen sich mit der Include-Anweisung einbinden, z.B.:

```
#include "admins.ini"
```

Steht dieser Eintrag in der users.ini, sucht Admin Mod auch noch in der admins.ini (im gleichen Verzeichnis) nach weiteren Usern (vergl. plugin.ini).

4.4.1.4. Serverseitige Einstellungen (adminmod.cfg)

Nun sollte noch einen Blick in die adminmod.cfg („addons/adminmod/config“) geworfen werden. Hier sollte die Variable password_field nach den eigenen Vorstellungen verändert werden. Diesen Variablennamen wird der Server vom Spielerconnect überprüfen. Es ist zu beachten, dass der Unterstrich zwingend vor den Variablennamen zu schreiben ist. Ansonsten wird der Server aus Sicherheitsgründen nicht starten. Der Eintrag sieht standardmäßig folgendermaßen aus:

```
password_field "_pw-home"
```

[3] http://forum.adminmod.de/info.php?go=tools.html

Der Hintergrund für die zwingend vorgeschriebene Nutzung des Unterstriches ergibt sich aus einem „Exploit“, mit dem jeder Admin das Passwort eines Users auslesen kann. Während die über setinfo gesetzten, userspezifischen Variablen ohne den Unterstrich mit der Serverfunktion `condump` ausgelesen werden können, ist dies beim vorangestellten Unterstrich nicht der Fall. Falls man also trotz mehrfacher Warnung das Passwort doch in der config.cfg oder autoexec.cfg setzt, kann ein böswilliger Admin eines anderen Servers nicht an die Variable herankommen.

Fortgeschrittene Methoden und Einstellungen in der adminmod.cfg:

default_access
admin_highlander
amv_private_server
reserve_slots
reserve_slots_msg
reserve_type
use_regex

4.4.2. Clientseitige Einstellungen

Im Kapitel 4.4.1 wurde beschrieben, wie dem Server das Wissen vermittelt wird, wer Admin ist und wer nicht. Es könnte aber jeder mit dem passenden Namen, ID oder IP kommen und behaupten er/sie wäre Admin. Admin Mod überprüft, sofern in der users.ini angegeben, daher zusätzlich das Passwort. Es gibt dabei zwei Möglichkeiten dieses dem Server zu übergeben.

`admin_login <Passwort>`

Mit diesem Befehl wird dem Server via Console das eigene Admin Mod-Passwort übergeben. Die Methode funktioniert nicht bei Namensreservierung und für die Slotreservierung, da man vor der Möglichkeit der Eingabe des Befehls bereits vom Server geworfen wird.

`setinfo "<password_field>" "<Passwort>"`

Die setinfo Variante ist die elegantere Methode zum Anmelden bei Admin Mod und die einzige bei Namens- und Slotreservierung. Man definiert eine Variable beim Client, in der das Passwort hinterlegt wird. Mehr zu dem Thema setinfo ist aus der FAQ zu entnehmen. Ein typischer Eintrag lautet daher:

`setinfo "_pw-home" "xxx"`

wobei xxx für das in der users.ini definierte Passwort steht. Falls man in der adminmod.cfg die Variable password_field verändert hat, muss man „_pw-home“ natürlich entsprechend abändern.

Merke: Auch bei verschlüsselten Passwörtern in der users.ini, muss beim Client IMMER das Passwort im Klartext angegeben werden.

Wo trage ich das ein? Viele Wege führen nach Rom:

1. Verknüpfung: Man erstellt eine Verknüpfung ausschließlich zum connecten auf den eigenen Server.

   Beispiel:

   ```
   X:\Steam\Steam.exe -applaunch 10 +exec adminpass.cfg
   +connect 213.239.218.84:27015
   ```

   wobei in adminpass.cfg (im Modverzeichnis, bsp. hier cstrike) die setinfo-Zeile für das Passwort steht. -applaunch 10 steht für Counter-Strike. Andere Mods haben andere Zahlen. Hat der Server ein Passwort, so muss +password <Serverpasswort> zusätzlich angefügt werden.

   Man kann das alles aber auch in der adminpass.cfg ausführen lassen, die dann in etwa so aussähe:

   ```
   setinfo "_pw-home" "xxx"
   password <Serverpasswort>
   rcon_password <Rcon-Passwort>
   connect 213.239.218.84:27015
   ```

   In die Verknüpfung müsste dann nur stehen:

   ```
   X:\Steam\Steam.exe -applaunch 10 +exec adminpass.cfg
   ```

2. Externe Tools: HLSW beispielsweise ermöglicht das Setzen eines Passworts beim Connect aus dem Programm heraus. Hierbei muss in den Servereigenschaften bei „Zusätzliche Parameter“ stehen:

   ```
   +exec adminpass.cfg
   ```

3. autoexec.cfg: Wird immer gerne genannt, funktioniert auch gut, stellt aber ein Sicherheitsrisiko dar, da das Passwort jedem Server mitgeteilt wird, nicht nur dem eigenen.

4. valve.rc: Das Selbe wie mit der autoexec.cfg.

Wofür man sich entscheidet, sei einem selbst überlassen. Es ist aber zu bedenken, dass die Optionen 3 und 4 ein erhebliches Sicherheitsrisiko bergen. Im Zweifel admin_login benutzen, sofern keine Reservierungen verwendet werden.

Wie erkennt man aber, dass der Server einen als Admin erkannt hat?

Die folgenden Zeilen müssen beim Connecten erscheinen:

```
[ADMIN] Checking password for user access...
[ADMIN] Password accepted for user '0815-Avg'. Access granted: 131071
```

Bei Steam ist nach dem Connect in die Console zu gehen und etwas scrollen, um die Zeilen zu finden.

Wird man dennoch nicht als Admin erkannt, sollte man zunächst die einzelnen Schritte

im Kapitel 4.4 genau durchgehen.

Es gibt allerdings drei besondere Punkte, die zu beachten sind.

1. Zunächst sollte die Variable encrypt_password überprüft werden. Diese muss passend zu den Passwörtern in der users.ini gesetzt sein. Wer keine Passwortverschlüsselung verwendet soll hier 0 nehmen.
2. Wenn use_regex verwendet wird (Variable auf 1), sollte man immer darauf achten, dass evtl. vorhandene Regex-Steuerzeichen „escaped“ werden.
3. Ebenfalls dürfte es Leuten mit einem Steam-Listenserver Probleme bereiten, IPs oder die STEAM_ID als Authentifizierungsmerkmal in der users.ini zu verwenden. Es geht jedoch über die Localhost Adresse (s. Kapitel 4.4.1.1).

4.4.3. Rechtelevel

Das Rechtesystem von Admin Mod basiert auf dem Dualsystem (2^x). Jedem Level ist ein Bit (x) zugeordnet. Allerdings wird nicht das Bit angegeben sondern die sich daraus ergebene Dezimalzahl (s. Tabellen 4.1 und 4.2 für die Standardplugins):

Beispiel: $2^4 = 16$ oder $2^8 = 256$

Binär entspricht 16 = 00001000 bzw. 256 = 10000000. Man erkennt in der Binärschreibweise die einzelnen gesetzten Rechtelevel. Jedem einzelnen Befehl wird in den Plugins ein Rechtelevel zugewiesen, den der Admin braucht, um den Befehl auszuführen. Da dies ganz unterschiedliche Rechtelevel sein können und man in der Regel mehrere Level gleichzeitig zuweisen möchte, muss man die einzelnen Rechtelevel addieren.

$$\sum_{i=0}^{n} x_i \cdot 2^i = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + ... + x_n \cdot 2^n$$

Beispiel: $2^4 + 2^8 = 16 + 256 = 272$ oder binär 10001000

Aus den Tabellen 4.1 und 4.2 können die Rechtelevel für die Befehle der Standardplugins entnommen werden. Die Rechtelevel reichen von 0 bis 65536. Es ist aber auch möglich höhere Werte zu verwenden (z.B. für admin_highlander oder Customplugins). Das Level 0 wird dem Spieler immer zugewiesen. Der Hauptadmin, der auch das Rcon-Passwort kennt, sollte fast alle Rechte erhalten:

$$\sum_{i=0}^{16} x_i \cdot 2^i - 2^{12} - 2^{14} - 2^{15} = 77823$$

Die abgezogenen Rechtelevel werden zwar hin und wieder von Customplugins für Befehle benutzt, sie haben jedoch eine besondere Bedeutung für Admin Mod.

4096 (2^{12}): Ein Admin mit diesem Level ist immun gegenüber anderen Admins (es gibt aber auch Plugins, die das umgehen). Die Immunität kann im Spielverlauf durch das Setzen von admin_ignore_immunity 1 ausgesetzt werden.

16384 (2^{14}): Der eigene Name bzw. die eigene ID oder IP ist damit reserviert und

Tabelle 4.1.: Rechtelevel für Befehle der Standardplugins (Teil 1)

Level	Befehle
0	admin_listmaps
	admin_messagemode
	admin_nextmap
	admin_nomessagemode
	admin_timeleft
	admin_userlist
	admin_version
	say currentmap
	say glow
	say nextmap
	say timeleft
1	admin_startvote
	admin_vote_kick
	admin_vote_map
	admin_vote_restart
	say mapvote
	say rockthevote
	say vote
2	admin_cancelvote
	admin_denymap
	admin_fraglimit
	admin_map
	admin_restart
	admin_restartround
	admin_startvote
	admin_timelimit
	say cancelvote
	say denymap
4	admin_reload
8	admin_pause
	admin_prematch
	admin_unpause
16	admin_nopass
	admin_pass
32	admin_balance
	admin_friendlyfire
	admin_gravity
	admin_restrict
	admin_restrictmenu
	admin_teamplay
	admin_unrestrict
64	admin_chat
	admin_csay
	admin_psay
	admin_say
	admin_ssay
	admin_tsay
	admin_vsay
128	admin_ct
	admin_kick
	admin_slap
	admin_slay
	admin_slayteam
	admin_t
256	admin_ban
	admin_banip
	admin_unban
512	admin_autokick
	admin_autoteambalance
	admin_buytime
	admin_c4timer
	admin_cfg
	admin_chattime
	admin_consistency
	admin_fadetoblack
	admin_flashlight
	admin_footsteps
	admin_forcecamera
	admin_forcechasecam
	admin_freezetime
	admin_ghostfrequency
	admin_hostname
	admin_hpenalty
	admin_kickpercent
	admin_limitteams
	admin_mapvoteratio
	admin_maxrounds
	admin_playerid
	admin_roundtime
	admin_servercfg
	admin_startmoney
	admin_tkpunish
	admin_weaponscheck
	admin_winlimit

Tabelle 4.2.: Rechtelevel für Befehle der Standardplugins (Teil 2)

Level	Befehle
1024	normalerweise nicht genutzt
2048	admin_gag admin_ungag
4096	Admin-Immunität
8192	admin_blue admin_bury admin_disco admin_fun admin_glow admin_godmode admin_green admin_listspawn admin_llama admin_movespawn admin_noclip admin_red admin_removespawn admin_spawn admin_stack admin_teleport admin_unbury admin_unllama admin_userorigin admin_yellow
16384	Nickreservierung
32768	Slotreservierung
65536	admin_execall admin_execclient admin_execteam admin_rcon

darf ausschließlich vom in der users.ini angegebenen Spieler mit dem passenden Passwort verwendet werden.

32768 (2^{15}): Man erhält einen exklusiven Zugang zum Server, selbst wenn dieser voll ist (Stichwort: reserve_type, reserve_slots). Es ist zu beachten, dass Admin Mod nicht zaubern kann; um auf den Server zu kommen ist immer noch ein freier Slot notwendig. Half-Life fängt bei vollem Server den Connectenden schon ab, bevor Admin Mod eingreifen kann.

Erst mit diesen drei gesonderten Leveln ergibt sich die häufig zu findende Empfehlung 131071, die aber meist gar nicht notwendig ist und oftmals mehr Probleme bereitet als hilft.

$\sum_{i=0}^{16} x_i \cdot 2^i = 131071$

4.4.4. Serverplätze reservieren

Wer kennt das nicht? Man möchte auf dem eigenen Server spielen, der aber voll ist. Dem Admin sollten besondere Rechte zugesprochen werden, dass er auch auf einen „vollen" Server connecten kann. Dazu wurden in Admin Mod die reservierten Plätze (oder auch Slots) eingeführt.

Um diese Funktion zu nutzen, ist etwas Vorarbeit zu leisten. Zunächst muss den Admins, die einen reservierten Slot nutzen dürfen, der Rechtelevel 32768 hinzugefügt werden (sofern nicht schon geschehen). Des Weiteren müssen sich entsprechende Admins mit der setinfo-Zeile anmelden. Und letztlich muss auch der Server für die Nutzung der Slots konfiguriert werden. Dabei sind die Anzahl der reservierten Slots (reserve_slots), die von den öffentlich verfügbaren Slots abgezogen werden (können nicht von normalen Spielern genutzt werden) und die Art der Reservierung (reserve_type) festzulegen. Optional kann auch eingestellt werden, welche Nachricht ein normaler Spieler erhält, wenn er versucht auf den Server zu kommen und nur reservierte Plätze erhältlich sind (reserve_slots_msg).

Während sich die Einstellungen reserve_slots und reserve_slots_msg noch von selbst erschließen, ist die richtige Einstellung von reserve_type nicht so einfach ersichtlich. Im weiteren sollen die drei vorhandenen Konzepte 0, 1 und 2 näher erläutert werden.

4.4.4.1. reserve_type 0

Dazu nehmen wir einen Server mit 14 Slots an und in Admin Mod wurden 2 Slots reserviert. Es sind somit 12 Slots öffentlich für alle Spieler verfügbar und 2 ausschließlich für Admins. Außerdem befinden sich zunächst 11 normale Spieler auf dem Server, die damit 11 der 12 öffentlichen Slots beanspruchen.

Bei der Einstellung 0 würde sowohl ein zusätzlicher normaler Spieler als auch ein Admin

den letzten öffentlichen Platz einnehmen. Sofern niemand vom Server geht, können nur noch Admins connecten. Kommt ein weiterer Admin, nimmt dieser einen der reservierten Slots in Anspruch. Geht nun ein Spieler oder Admin aus den öffentlichen Slots, rutscht der Admin im reservierten Slot in den freien öffentlichen. Es sind somit wieder alle öffentlichen Slots besetzt und die 2 reservierten frei (s.a. Abbildung 4.1).

Die Einstellung 0 hat den Vorteil, dass sofern nicht der Server voll ist, immer Admins auf den Server kommen können. Falls nicht alle öffentlichen Plätze belegt sind, nehmen die Admins jedoch den normalen Spielern die Plätze weg. Im schlimmsten Fall belegen die Admins alle öffentlichen Plätze, so dass die volle Slotzahl nicht ausgenutzt werden kann. Sind neben den öffentlichen auch alle reservierten Plätze besetzt, kann kein weiterer Admin auf den Server. Vorzugsweise sollte man nur einen Slot reservieren, da man ansonsten nur in seltenen Fällen wirklich mit mehr Spielern als der Anzahl der öffentlichen Plätze spielen kann.

4.4.4.2. reserve_type 1

Einen Slot ausschließlich zum Connecten der Admins zu verwenden, ist von den meisten gewünscht. Bei dieser Einstellung wird reserve_slots ignoriert und nur ein Slot reserviert. Der Platz bleibt immer frei, steht also nicht zum Spielen zur Verfügung. Er ist nur temporär für die Zeitdauer des Connectens für die Admins nutzbar. Für den Admin wird der Spieler mit dem höchsten Ping gekickt, und er gelangt in den frei gewordenen öffentlichen Slot (s.a. Abbildung 4.1). Dabei ist zu beachten, dass solange der connectende Admin nicht ins Spiel eingestiegen ist, er den Slot für weitere Admins blockiert. Diese Einstellung hat den großen Charme, dass man als Admin stets auf den Server kann, allerdings steht dauerhaft ein Slot weniger für alle Spieler zur Verfügung. Dies ist die beliebteste Einstellung.

4.4.4.3. reserve_type 2

Unter Annahme des Beispiels aus „reserve_type 0“ würde ein Admin bei der Einstellung 2 vorzugsweise einen reservierten Platz einnehmen. Sind 3 Admins auf dem Server, alle 14 Plätze sind belegt und ein Admin, der einen reservierten Slot hatte, geht vom Server, bekommt der Admin auf dem öffentlichen Platz den reservierten zugewiesen. Es ist dann also wieder ein öffentlicher Platz frei (s.a. Abbildung 4.1).

Die Einstellung 2 hat den Vorteil, dass, sofern nicht der Server voll ist, immer Admins auf den Server kommen können. Allerdings kann man die volle Slotanzahl auf dem Server nur nutzen, wenn auch mindestens soviele Admins wie reservierte Plätze spielen. Ist der Server voll, kann auch kein Admin mehr connecten. Die Einstellung lohnt sich, wenn wenige Admins häufig spielen.

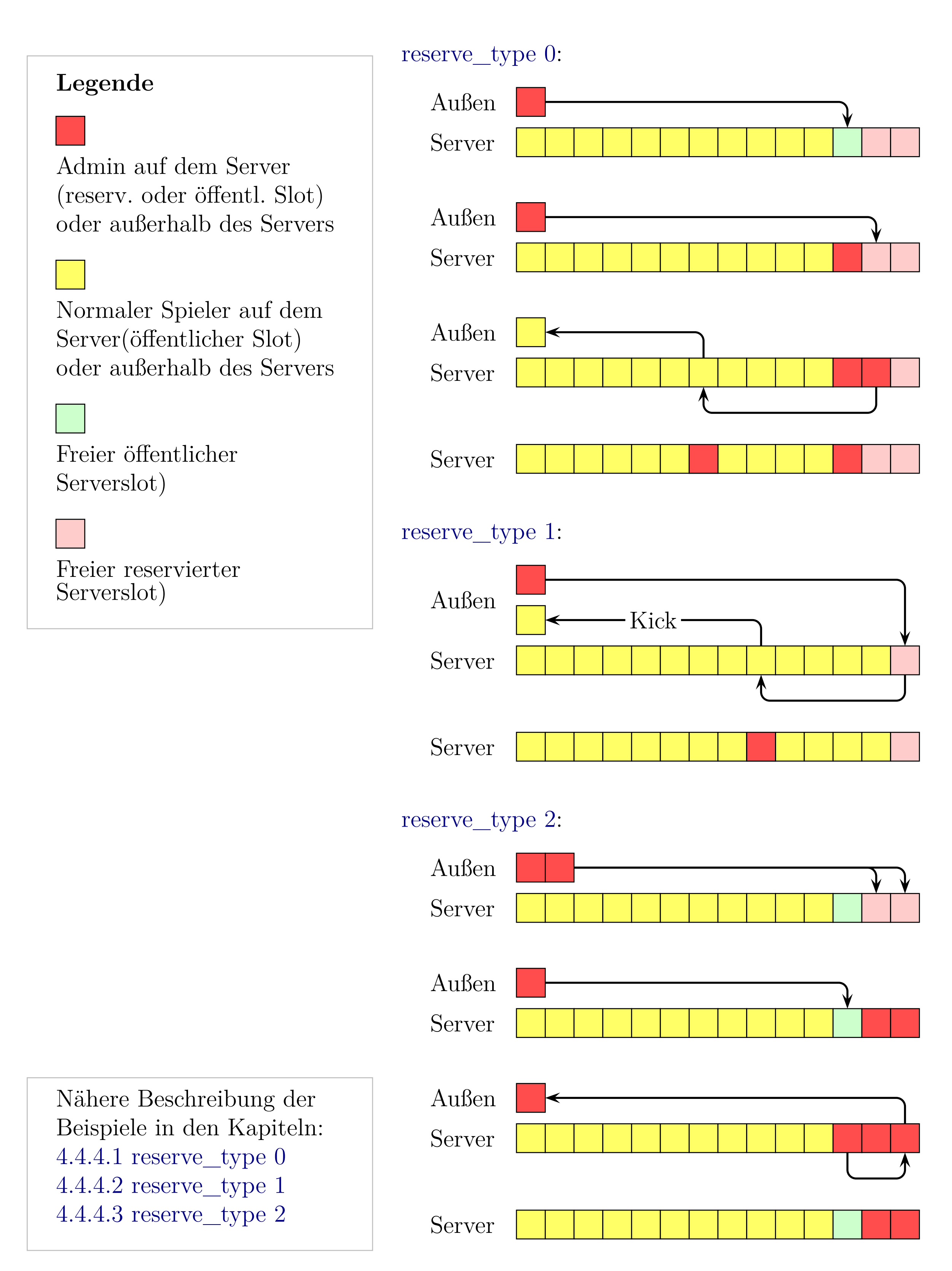


Abbildung 4.1.: Beispiele für reserve_type 0, 1 und 2

4.4.5. Namen reservieren

Es hindert niemanden auf dem Server den Namen eines Admins anzunehmen. Auch wenn er dadurch keine Rechte auf dem Server bekommt, könnte der entsprechende Spieler mit der Identität des Admins durch auffälliges Verhalten dessen Ruf verschlechtern. Dafür bietet Admin Mod die Möglichkeit den eigenen Namen zu reservieren.

Folgender Eintrag in der users.ini verhindert, dass jemand den eigenen Namen verwendet:

```
STEAM_0:1:9174:meinpasswort:94207
[WING] Black Knight:zufälligespasswort:16384
```

Der erste Eintrag ist der übliche Eintrag mit einer Steam-ID. Der zweite Eintrag reserviert mit dem Recht 16384 den Namen „[WING] Black Knight“ mit einem zufällig ausgesuchten Passwort. Das Passwort kann man wählen, wie man will. Wichtig ist, dass es keiner erraten kann. Jeder, der den Namen annehmen will, wird vom Server gekickt. Man beachte aber, dass der zugehörige Accesslevel im ersten Eintrag das Recht 16384 beinhalten muss. Sollte dies nicht der Fall sein, sucht Admin Mod nach der erfolgreichen Erkennung als Admin weiter und wirft den Admin anschließend vom Server, da er das zufällige Passwort nicht gesetzt hat und auch nicht setzen kann.

Das Ganze kann man für alle Clanmitglieder wiederholen, was aber sehr arbeitsintensiv sein sein kann. Eleganter erreicht man das, indem man mittels RegEx gleich den Clantag reserviert, Abschnitt 4.4.6.

4.4.6. RegEx (Clantags reservieren etc.)

4.4.6.1. Einführung

Was sind Regular Expressions (RegEx)? Der deutsche Begriff „reguläre Ausdrücke“ ist zunächst ebenfalls nicht weiterführend.

Mit regulären Ausdrücken kann man nach Mustern in einer Zeichenkette (String) suchen. Der RegEx ist dabei das Muster. Das Muster muss dabei nicht aus einer Zeichenabfolge bestehen sondern kann auch sogenannte Steuerzeichen beinhalten. Diese Steuerzeichen sind ganz normale Zeichen, die RegEx aber als Befehl interpretiert.

Vom Betriebssystem kennt man das Globbing. Dieses besitzt zwei Steuerzeichen. Zum einen ist das der Asterisk „*“ für eine Zeichenkette beliebiger Länge und das Fragezeichen „?“ für eine beliebige Zeichenkette der Länge 1.

Sucht man beispielsweise nach einer Datei mit der Endung „.exe“, gibt man in der Suche „*.exe“ ein. Sucht man hingegen alle Dateien mit der Endung „.exe“ und nur einem Buchstaben vor dem Suffix (z.B. „a.exe“), dann gibt man „?.exe“ an.

RegEx nutzt ebenfalls Steuerzeichen. Allerdings sind das deutlich mehr und andere, bzw. sie haben eine andere Bedeutung.

Der Punkt „." ist im RegEx vergleichbar mit dem „?" beim Globbing. „.*" hingegen bedeutet, dass beliebig viele oder aber auch keine Zeichen gefunden werden. Dies entspricht dem Asterisk beim Globbing.

Will man, wie im oberen Globbing-Beispiel, nach Dateien mit der Endung „.exe" suchen, so hätte man das Problem, dass der Punkt ja jedes beliebige Zeichen bedeuten kann. Man muss also kenntlich machen, dass es sich beim Punkt diesmal nicht um ein Steuerzeichen handelt. Dazu muss man den Punkt „escapen". Dies wird mit einem vorangestellten Backslash „\„ erreicht. Ein möglicher RegEx lautet demnach „.*\.exe".

Weitere interessante Steuerzeichen wären „^" und „$".

„^" bedeutet, dass der nachfolgende RegEx am Anfang der durchsuchten Zeichenkette stehen muss. „^Admi" wird in „Admin Mod" gefunden „^od" jedoch nicht, da „od" nicht am Anfang der Zeichenkette steht.

„$" hingegen macht das Gleiche für das Ende der Zeichenkette. „Admi$" wird nicht in „Admin Mod" gefunden, „od$" jedoch sehr wohl, da es am Zeichenkettenende steht.

Die eckigen Klammer („[„ und „]") geben eine Optionsmenge an. „[Aa]" bedeutet, dass entweder ein großes oder ein kleines „A" gesucht wird. Wendet man dieses RegEx auf das obige Beispiel „Admin Mod" an, so würde es gefunden, während ein „a" keine Übereinstimmung findet. Man kann es aber auch negieren, also nach Zeichen suchen, die nicht vorhanden sein sollen („^[Aa]"). Man kann aber auch Bereiche angeben, z.B. [a-z] oder [0-9], alle Kleinbuchstaben oder die Zahlen von 0-9. Bei ersterem werden Umlaute nicht erfasst.

Admin Mod benutzt Extended (oder Modern) Regular Expressions nach POSIX 1003.2. Diese werden jedoch case-insensitive gesucht (Groß- und Kleinschreibung wird nicht unterschieden).

Weitere und ausführlichere Informationen zu RegEx sind im Internet zu finden[4].

4.4.6.2. Clantag reservieren

Ein Clantag ist nur ein Teil des eigenen Namens. Er weist einen jedoch indirekt als Mitglied des Clans und auf dem eigenen Server auch als Admin aus. Es ist dementsprechend auch nicht gewollt, dass ein Nicht-Mitglied so tut, als ob er Mitglied bzw. Admin auf dem Server ist.

Einerseits kann man wie im vorherigen Abschnitt 4.4.5 beschrieben, den Namen jedes einzelnen Spielers inklusive Clantag reservieren. Dies ist auf der einen Seite unbequem und hat zudem den Nachteil, dass Kombinationen mit dem Clantag und nicht reservierten Namen möglich sind. Es lohnt sich daher für die meisten nur den Clantag zu reservieren.

Zunächst sind RegEx in der adminmod.cfg zu aktivieren (use_regex 1) und alle Spieler in

[4] z.B. http://www.regenechsen.de/phpwcms/index.php?regex_dt_tb

die users.ini einzutragen. Als LETZTEN Eintrag schreibt man den Clantag mit einem x-beliebigen Passwort und dem Recht 16384 zum Reservieren dieses Namens. Das Passwort sollte auf jeden Fall von niemanden zu erraten sein. Es wird auch von keinem Admin gebraucht.

Jeder Admin in der users.ini, der Clanmitglied ist, muss mit dem Recht 16384 (Namensreservierung) ausgestattet sein. Anderenfalls wird er bei Verwendung des Clantags gekickt. Admin Mod scannt die users.ini nach der Adminerkennung weiter.

Der Clantag-Eintrag muss der letzte Eintrag sein. Sollten weitere Clanmitglieder in der users.ini folgen, würden sie bereits von der Clantagerkennung gekickt werden, bevor sie sich als Admin authentifizieren können.

Clantags bestehen meist aus diversen Sonderzeichen, die der RegEx-Algorithmus als Steuerzeichen ansieht, d.h. es kommt zu einer kompletten Missinterpretation. Daher muss jedes Sonderzeichen „escaped“ werden. Dies geschieht über einen Backslash „\„. Im Falle des Clans [WING] sind die eckigen Klammern ein Problem. Daher müsste man zum schützen des Clantags z.B. schreiben:

```
\[WING\]:sdjgfsjg:16384
```

„Escaped“ werden müssen folgende Zeichen „*, +, ?, ., (,), [,], {, }, \, |, ^, $“.

Verwendet man zusätzlich weitere Namen in der users.ini, müssen auch die dort verwendeten Sonderzeichen „escaped“ werden. Dies gilt nicht für IDs oder IPs. Hier erkennt Admin Mod, dass es sich nicht um Namen handelt und wendet RegEx nicht an.

4.5. Benutzbare Maps einstellen (maps.ini)

Die maps.ini schränkt die Benutzung von Maps mittels admin_map oder Mapvotes ein. Ausschließlich Maps, die in dieser Datei stehen, dürfen dann benutzt werden. Anderenfalls bekommt man bei admin_map o.ä. die Rückmeldung „bad mapname“.

Die Verwendung einer solchen Datei ist jedoch OPTIONAL. Um die Funktion abzuschalten setzt man:

```
maps_file 0
```

Will man hingegen die Verwendung bestimmter Maps einschränken, schreibt man:

```
maps_file "addons/adminmod/config/maps.ini"
```

Wahlweise kann man natürlich auch einen anderen Pfad oder Dateinamen wählen.

Die Maps werden genau wie in der mapcycle.txt Datei aufgeführt, z.B.:

```
de_dust
de_inferno
de_dust2
cs_office
```

Daraus ergibt sich natürlich auch eine nette Möglichkeit die benutzbaren Maps auf die im Mapcycle zu beschränken. Man muss lediglich den aktuellen Mapcycle als maps.ini definieren:

```
maps_file "mapcycle.txt"
```

Das Einbinden weiterer Dateien mittels Include-Anweisung ist leider nicht möglich.

4.6. IPs für reservierte Slots festlegen (ips.ini)

Die ips.ini beinhaltet IP-Adressen, die stets Zugang zu reservierten Slots haben (siehe auch: reserve_type und reserve_slots) und ist OPTIONAL. Über die users.ini lassen sich zwar einzelnen IPs reservierte Slots zuweisen, allerdings ist es dort nicht möglich IP-Bereiche festzulegen.

Interessant dürfte eine solche Datei für Internetcafes sein, die in ihrem Netz einen Gameserver mit Internetanschluss betreiben. Den eigenen Rechnern möchte man natürlich Vorrang geben, ohne gleich Adminrechte zu vergeben. Genauso könnte man auch einen HLTV den Zugang zu einem vollem Server ermöglichen.

Für die meisten Spieler im Internet ist die IP jedoch nicht nützlich, da sie dynamisch bei jeder Einwahl vergeben wird, man also stets eine neue IP erhält.

Die ips.ini wird über ips_file definiert; in der Regel schreibt man das folgendermaßen:

```
ips_file "addons/adminmod/config/ips.ini"
```

Es ist natürlich auch jeder andere relative Pfad oder Dateiname möglich. Will man die Datei nicht benutzen, so schreibt man:

```
ips_file 0
```

Einzelne IPs trägt man folgendermaßen in die Datei ein:

```
129.49.231.126
131.112.1.12
71.17.235.3
172.54.512.7
```

Nutzt man Subnets, so kann man auch das berücksichtigen lassen:

```
129.49.231.126/255.255.255.0
```

Um beispielsweise die IPs von 168.23.21.0 bis 168.23.21.255 für die reservierten Slots zu erlauben, schreibt man den Bereich mit einer 0 am Ende:

```
168.23.21.0/255.255.255.0
```

Das kann man natürlich auch auf größere Netzwerke ausweiten:

```
168.23.0.0/255.255.0.0
```

Weitere Dateien lassen sich mit der Include-Anweisung einbinden, z.B.:

```
#include "ipadmins.ini"
```

Steht dieser Eintrag in der ips.ini, sucht Admin Mod auch noch in der ipadmins.ini (im gleichen Verzeichnis) nach weiteren IPs (vergl. plugin_ini).

Änderungen an der ips.ini werden erst bei einem Serverneustart, einem Mapwechsel oder vorzugsweise beim Ausführen des Befehls admin_reload übernommen.

4.7. Models reservieren (models.ini)

Die models.ini ist eine OPTIONALE Datei. Sie kann bestimmte Models mittels Passwort vor der Benutzung durch andere Spieler schützen.

Sie wird mittels der Variablen models_file gesetzt, z.B.:

```
models_file "addons/adminmod/config/models.ini"
```

Der relative Pfad sowie der Dateiname können frei gewählt werden.

Um den Modelschutz abzuschalten schreibt man hingegen:

```
models_file 0
```

Einträge in die models.ini werden nach folgendem Muster durchgeführt:

```
model:passwort
```

Das Passwort ist das gleiche, das auch in der users.ini beim entsprechenden Spieler vermerkt ist. Bei Verwendung von verschlüsselten Passwörtern in der users.ini müssen diese in der models.ini ebenfalls verschlüsselt angegeben werden.

Beispiele:

```
sas:meinampasswort
```

Limitiert die Benutzung des CT-Models „sas“ aus Counter-Strike auf Spieler mit dem Passwort „meinampasswort“ in der users.ini.

```
hostage1:gsddkakfla
hostage2:gsddkakfla
hostage3:gsddkakfla
```

Verhindert das Benutzen von Hostage-Models zum Cheaten in CS. Das Passwort muss irgendetwas nicht zu Erratenes sein. Vermutlich besteht die Cheat-Möglichkeit inzwischen nicht mehr.

Der Nutzen dieser Datei ist eingeschränkt. Sollen Spieler mit unterschiedlichen Passwörtern in der users.ini das gleiche Model reserviert bekommen, wird dies nicht funktionieren, da nur ein Passwort dem Model zugeordnet werden kann. In Mods mit wenigen Models (insbesondere in Teamspielen wie Counter-Strike oder Day of Defeat) ist die Verwendung

der models.ini daher nicht sinnvoll. Haben alle Admins in der users.ini jedoch das gleiche Passwort (bei Verwendung der Steam-ID durchaus akzeptabel), kann auch dem Clan ein Model zugeordnet werden.

Ein weiterer Punkt ist, dass ein Spieler bei Auswahl eines geschützten Models direkt vom Server gekickt wird. Dies ist eventuell nicht unbedingt im Sinne vieler Admins.

Weitere Dateien lassen sich mit der Include-Anweisung einbinden, z.B.:

```
#include "modelsadmins.ini"
```

Steht dieser Eintrag in der models.ini, sucht Admin Mod auch noch in der modelsadmins.ini (im gleichen Verzeichnis) nach weiteren Models (vergl. plugin.ini).

Änderungen an der models.ini werden erst bei einem Serverneustart, einem Mapwechsel oder vorzugsweise beim Ausführen des Befehls admin_reload übernommen.

4.8. Chat zensieren (wordlist.txt)

Es gibt eine Möglichkeit mit Admin Mod zu verhindern, dass bestimmte Worte auf dem Server im Chat geschrieben werden. Beschimpfungen und Ähnliches können zensiert werden.

Dies geschieht über das Plugin Retribution, welches zu den Standardplugins gehört, und der wordlist.txt Datei.

In die wordlist.txt schreibt man einfach die Wörter untereinander, die nicht im Chat auftauchen sollen:

```
Idiot
Nap
Naps
```

Aktiviert wird die Funktion über die adminmod.cfg beispielsweise folgendermaßen:

```
words_file "addons/adminmod/config/wordlist.txt"
```

Deaktiviert wird die Zensur wie folgt:

```
words_file 0
```

Die Liste wird case-insensitive (also ohne Unterschied zwischen Groß- und Kleinschreibung) überprüft. Änderungen werden erst nach einem Serverrestart, einem Mapwechsel oder nach dem Aufruf von admin_reload aktiv.

Allerdings können bereits kleinste Änderungen die Liste umgehen:

```
N.aps
```

Dies ist immer noch lesbar, aber wird nach obiger Liste auf Grund des zusätzlichen Punktes nicht als Beschimpfung erkannt.

4.9. Pluginspezifische Einstellungen (vault.ini)

Die vault.ini dient der Speicherung von Plugineinstellungen. Insbesondere Customplugins machen oftmals Gebrauch davon.

Grundsätzlich sollte ein guter Pluginauthor dem User über Befehle im Plugin die Einstellungen ohne direkten Zugriff ermöglichen. Manchmal muss man aber auch manuell Hand anlegen.

Aktiviert wird die vault.ini in der adminmod.cfg:

`admin_vault_file "addons/adminmod/config/vault.ini"`

Zum Abschalten der vault.ini trägt man ein:

`admin_vault_file 0`

Das ist allerdings auf Grund der oftmaligen Nutzung nicht zu empfehlen.

Manuelle Änderungen sollten nicht während des Serverbetriebes vorgenommen werden. Sollte kurz nach der Änderung ein Plugin in die vault.ini schreiben, so sind die Änderungen verschwunden. Die gesamten Einstellungen werden nämlich von Admin Mod beim Mapstart in den Speicher geladen und stets am Stück zurückgeschrieben. D.h. bei manueller Änderung gelangt diese nicht sofort in den Speicher. Dies kann man nur über einen Mapchange oder admin_reload erreichen.

4.10. MySQL Installation einrichten

Es wird davon ausgegangen, dass Admin Mod bereits in einer Fallback-Konfiguration (Benutzung von Dateien) läuft. Eine Anleitung ist in Kapitel 3.4 zu finden.

In Admin Mod-MySQL können die Dateien users.ini, models.ini, ips.ini, wordlist.txt und plugin.ini durch Tabellen ersetzt werden. Es kann aber auch eine Mischinstallation durchgeführt werden. In der Regel wird nur die users und tags Tabelle benötigt. Besonderes Augenmerk sollte man auf die users.ini haben. Diese wird aufgeteilt in eine users- und eine tags-Tabelle. Admin Mod bietet die Möglichkeit über RegEx den eigenen Clantag zu schützen. Da Regex unter SQL ein sehr rechenintensiver Befehl ist, wenn er auf große Datenbestände losgelassen wird, empfiehlt es sich die Benutzung auf eine kleine überschaubare Tabelle zu limitieren, die nur die zu schützenden Clantags beinhaltet.

Auch wenn Tabellen genutzt werden, ist es immer gut auch eine einfache funktionierende Konfiguration basierend auf Dateien zu besitzen, da Admin Mod bei Verbindungsverlust zum MySQL-Server dann auf diese zurückgreifen kann.

Im Weiteren wird die Einrichtung der Tabellen in der MySQL-Datenbank näher beleuchtet.

4.10.1. Datenbankeinrichtung

Sofern noch keine Datenbank für Admin Mod eingerichtet wurde, sollte man dies tun.

```
mysql> CREATE DATABASE adminmod;
```

Als Datenbanknamen kann man statt „adminmod“ auch jeden anderen Namen angeben. Admin Mod kann aber auch in jede bestehende Datenbank eingebaut werden. Dabei sind allerdings eventuelle Überschneidungen bei Tabellennamen zu berücksichtigen und anzupassen. Der Datenbankname ist unter mysql_database in der adminmod.cfg anzugeben.

4.10.2. Users-Tabelle

Die Users-Tabelle wird als Ersatz für die users.ini eingerichtet. Sie besteht aus den Spalten „nick“, „pass“ und „access“. Äquivalent zur Angabe in der users.ini handelt es sich dabei um den Usernamen oder ID, das zugehörige Passwort und den Rechten des Users.

```
mysql> CREATE TABLE users( nick VARCHAR(31) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL, access INTEGER UNSIGNED NOT NULL);
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_users angegeben werden. Es ist daher auch möglich einen anderen Namen für die Users-Tabelle zu wählen.

4.10.3. Tags-Tabelle

Wie bereits beschrieben ist die Verwendung von RegEx innerhalb großer Datenbeständen ineffizient. Daher sollten zu schützende Clantags in diese Extratabelle geschrieben werden. Diese sollte dadurch erheblich kleiner ausfallen. Angelegt wird sie natürlich genauso wie die Users-Tabelle.

```
mysql> CREATE TABLE tags( tag VARCHAR(30) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL, access INTEGER UNSIGNED NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_tags angegeben werden. Es ist daher auch möglich einen anderen Namen für die Tags-Tabelle zu wählen.

4.10.4. Models-Tabelle

Die Models-Tabelle wird als Ersatz für models.ini eingerichtet. Sie besteht aus den Spalten „nick“ und „pass“. Äquivalent zur Angabe in der models.ini handelt es sich dabei bei „nick“ um den Modelnamen und bei „pass“ um das Passwort.

```
mysql> CREATE TABLE models( nick VARCHAR(20) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_models angegeben werden. Es ist daher auch möglich einen anderen Namen für die Models-Tabelle zu wählen.

4.10.5. Ips-Tabelle

Die Ips-Tabelle wird als Ersatz für ips.ini eingerichtet. Sie besteht nur aus der Spalte „ip“. Einträge in diese Tabelle sind identisch zur ips.ini.

```
mysql> CREATE TABLE ips( ip VARCHAR(22) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_ips angegeben werden. Es ist daher auch möglich einen anderen Namen für die Ips-Tabelle zu wählen.

4.10.6. Words-Tabelle

Die Words-Tabelle wird als Ersatz für wordlist.txt eingerichtet. Sie besteht nur aus der Spalte „word“. Einträge in diese Tabelle sind identisch zur wordlist.txt.

```
mysql> CREATE TABLE words( word VARCHAR(30) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_words angegeben werden. Es ist daher auch möglich einen anderen Namen für die Words-Tabelle zu wählen.

4.10.7. Plugins-Tabelle

Die Plugins-Tabelle wird als Ersatz für plugin.ini eingerichtet. Sie besteht nur aus der Spalte „plugin“. Einträge in diese Tabelle sind identisch zur plugin.ini.

```
mysql> CREATE TABLE plugins( plugin VARCHAR(100) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter mysql_dbtable_plugins angegeben werden. Es ist daher auch möglich einen anderen Namen für die Plugins-Tabelle zu wählen.

4.10.8. Zugriff auf die Datenbank

Nachdem die MySQL-Datenbank nun mit den Tabellen ausgestattet ist, muss Admin Mod noch mitgeteilt bekommen, wo und wie es diese erreichen kann. Dazu müssen in der adminmod.cfg folgende Variablen gesetzt sein:

mysql_host (IP-Adresse Datenbankrechner, meist „localhost“)
mysql_user (Username zum Zugriff auf die Datenbank)
mysql_pass (Passwort zum Zugriff auf die Datenbank)

4.10.9. Außergewöhnliche Einstellungen

Es gibt noch Einstellungen, die nur sehr selten gebraucht werden, die aber erwähnt werden sollten.

mysql_preload legt fest, ob alle Spieler beim Mapwechsel in den Speicher von Admin Mod geladen werden (Einstellung: 1) oder aber bei jedem Spielerconnect eine Datenabankabfrage durchgeführt wird (Einstellung: 0). Ersteres lohnt sich bei „wenigen“ Spielern und/oder nur wenigen Änderungen in der Datenbank. Letzteres ist die flexiblere Lösung, da Änderungen in der Datenbank sofort beim nächsten Connect aktiv sind. Häufige Datenbankabfragen müssen jedoch in Kauf genommen werden, was ggf. für Verzögerungen im Spielverlauf (Lags) sorgen kann. Dies ist aber eine sehr hypothetische Einschränkung, so dass man stets den Preload abschalten sollte. Sollte es Probleme geben, kann man immer noch wechseln.

Schaltet man den Preload der Daten ab, eröffnet sich einem die Möglichkeit auch nicht standardkonforme SQL-Abfragen durchzuführen. Dies geschieht mittels der Variablen:

mysql_users_sql
mysql_tags_sql

Die Verwendung der Variablen ist nicht trivial, so dass nur Profis und Frustrationstolerante sich daran versuchen sollten. Wer den Preload weiterhin verwenden möchte oder sich nicht so mit der von Admin Mod erwarteten Syntax auskennt, der kann sich ggf. mit MySQL-Views (Version 5.0.1 oder neuer notwendig) behelfen. Auf die einzelnen Möglichkeiten soll im Abschnitt 4.12 am Beispiel der Einbindung Admin Mods in die beliebte Forumsoftware phpBB[5] eingegangen werden.

4.11. PostgreSQL Installation einrichten

Es wird davon ausgegangen, dass Admin Mod bereits in einer Fallback-Konfiguration (Benutzung von Dateien) läuft. Eine Anleitung ist in Kapitel 3.4 zu finden.

In Admin Mod-PostgreSQL können die Dateien users.ini, models.ini, ips.ini, wordlist.txt und plugin.ini durch Tabellen ersetzt werden. Es kann aber auch eine Mischinstallation durchgeführt werden. In der Regel wird nur die users und tags Tabelle benötigt. Besonderes Augenmerk sollte man auf die users.ini haben. Diese wird aufgeteilt in eine users- und eine tags-Tabelle. Admin Mod bietet die Möglichkeit über RegEx den eigenen Clantag zu schützen. Da Regex unter SQL ein sehr rechenaufwendiger Befehl ist, wenn er auf große Datenbestände losgelassen wird, empfiehlt es sich die Benutzung auf eine kleine überschaubare Tabelle zu limitieren, die nur die zu schützenden Clantags beinhaltet.

Auch wenn Tabellen genutzt werden, ist es immer gut auch eine einfache funktionierende Konfiguration basierend auf Dateien zu besitzen, da Admin Mod bei Verbindungsverlust

[5] http://www.phpBB.com/

zum PostgreSQL-Server auf diese dann zurückgreifen kann.

Im Weiteren wird die Einrichtung der Tabellen in der PostgreSQL-Datenbank näher beleuchtet.

4.11.1. Datenbankeinrichtung

Sofern noch keine Datenbank für Admin Mod eingerichtet wurde, sollte man dies tun.

```
$ createdb adminmod;
```

Als Datenbanknamen kann man statt „adminmod“ auch jeden anderen Namen angeben. Admin Mod kann aber auch in jede bestehende Datenbank eingebaut werden. Dabei sind allerdings eventuelle Überschneidungen bei Tabellennamen zu berücksichtigen und anzupassen. Der Datenbankname ist unter pgsql_database in der adminmod.cfg anzugeben.

4.11.2. Users-Tabelle

Die Users-Tabelle wird als Ersatz für die users.ini eingerichtet. Sie besteht aus den Spalten „nick“, „pass“ und „access“. Äquivalent zur Angabe in der users.ini handelt es sich dabei um den Usernamen oder ID, das zugehörige Passwort und den Rechten des Users.

```
adminmod=> CREATE TABLE users( nick VARCHAR(31) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL, access INTEGER UNSIGNED NOT NULL);
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_users angegeben werden. Es ist daher auch möglich einen anderen Namen für die Users-Tabelle zu wählen.

4.11.3. Tags-Tabelle

Wie bereits beschrieben ist die Verwendung von RegEx innerhalb großer Datenbeständen ineffizient. Daher sollten zu schützende Clantags in diese Extratabelle geschrieben werden. Diese sollte dadurch erheblich kleiner ausfallen. Angelegt wird sie natürlich genauso wie die Users-Tabelle.

```
adminmod=> CREATE TABLE tags( tag VARCHAR(30) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL, access INTEGER UNSIGNED NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_tags angegeben werden. Es ist daher auch möglich einen anderen Namen für die Tags-Tabelle zu wählen.

4.11.4. Models-Tabelle

Die Models-Tabelle wird als Ersatz für models.ini eingerichtet. Sie besteht aus den Spalten „nick“ und „pass“. Äquivalent zur Angabe in der models.ini handelt es sich dabei bei „nick“ um den Modelnamen und bei „pass“ um das Passwort.

```
adminmod=> CREATE TABLE models( nick VARCHAR(20) PRIMARY KEY NOT NULL,
pass VARCHAR(64) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_models angegeben werden. Es ist daher auch möglich einen anderen Namen für die Models-Tabelle zu wählen.

4.11.5. Ips-Tabelle

Die Ips-Tabelle wird als Ersatz für ips.ini eingerichtet. Sie besteht nur aus der Spalte „ip“. Einträge in diese Tabelle sind identisch zur ips.ini.

```
adminmod=> CREATE TABLE ips( ip VARCHAR(22) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_ips angegeben werden. Es ist daher auch möglich einen anderen Namen für die Ips-Tabelle zu wählen.

4.11.6. Words-Tabelle

Die Words-Tabelle wird als Ersatz für wordlist.txt eingerichtet. Sie besteht nur aus der Spalte „word“. Einträge in diese Tabelle sind identisch zur wordlist.txt.

```
adminmod=> CREATE TABLE words( word VARCHAR(30) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_words angegeben werden. Es ist daher auch möglich einen anderen Namen für die Words-Tabelle zu wählen.

4.11.7. Plugins-Tabelle

Die Plugins-Tabelle wird als Ersatz für plugin.ini eingerichtet. Sie besteht nur aus der Spalte „plugin“. Einträge in diese Tabelle sind identisch zur plugin.ini.

```
adminmod=> CREATE TABLE plugins( plugin VARCHAR(100) NOT NULL );
```

Der Tabellenname muss in der adminmod.cfg unter pgsql_dbtable_plugins angegeben werden. Es ist daher auch möglich einen anderen Namen für die Plugins-Tabelle zu wählen.

4.11.8. Zugriff auf die Datenbank

Nachdem die PostgreSQL-Datenbank nun mit den Tabellen ausgestattet ist, muss Admin Mod noch mitgeteilt bekommen, wo und wie es diese erreichen kann. Dazu müssen in der adminmod.cfg folgende Variablen gesetzt sein:

pgsql_host (IP-Adresse Datenbankrechner, meist „localhost“)
pgsql_user (Username zum Zugriff auf die Datenbank)
pgsql_pass (Passwort zum Zugriff auf die Datenbank)
pgsql_port (Port zum Zugriff auf die Datenbank)

4.11.9. Außergewöhnliche Einstellungen

Es gibt noch Einstellungen, die nur sehr selten gebraucht werden, die aber erwähnt werden sollten.

pgsql_preload legt fest, ob alle Spieler beim Mapwechsel in den Speicher von Admin Mod geladen werden (Einstellung: 1) oder aber bei jedem Spielerconnect eine Datenabankabfrage durchgeführt wird (Einstellung: 0). Ersteres lohnt sich bei „wenigen“ Spielern und/oder nur wenigen Änderungen in der Datenbank. Letzteres ist die flexiblere Lösung, da Änderungen in der Datenbank sofort beim nächsten Connect aktiv sind. Häufige Datenbankabfragen müssen jedoch in Kauf genommen werden, was ggf. für Verzögerungen im Spielverlauf (Lags) sorgen kann. Dies ist aber eine sehr hypothetische Einschränkung, so dass man stets den Preload abschalten sollte. Sollte es Probleme geben, kann man immer noch wechseln.

Schaltet man den Preload der Daten ab, eröffnet sich einem die Möglichkeit auch nicht standardkonforme SQL-Abfragen durchzuführen. Dies geschieht mittels der Variablen:

pgsql_users_sql
pgsql_tags_sql

Die Verwendung der Variablen ist nicht trivial, so dass nur Profis und Frustrationstolerante sich daran versuchen sollten. Wer den Preload weiterhin verwenden möchte oder sich nicht so mit der von Admin Mod erwarteten Syntax auskennt, der kann sich ggf. mit PostgreSQL-Views behelfen.

4.12. Beispiel: Admin Mod in phpBB integrieren (MySQL)

In diesem Abschnitt soll die Einbindung Admin Mods in ein bestehendes Content Management System (CMS) oder Webforum mit MySQL-Backend[6] am Beispiel des populären

[6] http://www.mysql.com

Forums phpBB[7] beschrieben werden. Dies hat den Charme, dass man die Userverwaltung bequem aus der Forumssoftware heraus durchführen kann.

Die vorgestellten Methoden haben nicht den Anspruch perfekt zu sein. Ich bin weder gelernter Informatiker noch kenne ich mich besonders gut in MySQL aus. Vielmehr soll ein Einstieg gegeben werden, der zu Ideen und Optimierung anregen soll.

4.12.1. Admin Mod einrichten

Wie in Abschnitt Users-Tabelle gezeigt, erwartet Admin Mod standardmäßig eine Spalte für den Namen/ID (nick), das Passwort (pass) und den Accesslevel (access). Davon steht zunächst einmal das Passwort als MD5-Hash in phpBB2 zur Verfügung. phpBB3 weist einen gravierenden Unterschied zu phpBB2 auf. Die MD5-Hashes werden nicht mehr als MySQL-Hash sondern als „Salted Hash" (softwaremäßig nachverschlüsselt) gespeichert. Sie sind daher für Admin Mod nicht nutzbar. Dem entsprechend ist zunächst die Verschlüsselung (encrypt_password) richtig einzustellen:

phpBB2 (MD5-Hashes):
`encrypt_password 2`

phpBB3 (Plain-Text):
`encrypt_password 0`

Die weitere Beschreibung bezieht sich auf phpBB3. Im weiteren Verlauf des Kapitels werden jedoch einige Beispiele für phpBB2 gezeigt.

Die genutzte Tabelle muss in mysql_dbtable_users festgelegt werden und ist dabei in der Regel „phpbb_users":

```
mysql_dbtable_users "phpbb_users"
```

Nun stimmen die Spaltennamen nicht überein, so dass vom Standardformat abgewichen werden muss (mysql_users_sql). Die Abfrage würde also vorläufig folgendermaßen lauten:

```
SELECT user_password AS pass, 0 AS access FROM %s WHERE username='%s' OR
username='%s'
```

Es wird verglichen, ob der Nickname oder dessen Steam ID mit dem Nicknamen im Forum übereinstimmt. Ist das der Fall, wird das Passwort und der Accesslevel als „pass" bzw. „access" zurückgegeben.

Wie unschwer zu erkennen ist, hat phpBB kein Feld für den Accesslevel. In dieser Abfrage bekommen alle Spieler den Accesslevel 0. Mit phpBB3 ist es möglich userspezifische Profilfelder hinzuzufügen. Man fügt also ein INT-Feld („user_amaccess", wird dann in „pf_user_amaccess" umbenannt) hinzu, welches nur durch die Forenadministratoren editierbar ist (standardmäßig sollte es 0 oder 1 sein, s. default_access). Leider befindet

[7] http://www.phpbb.com

sich der Inhalt des Feldes nicht in „phpbb_users“, was die Abfrage komplexer macht. Auch muss man in phpBB3 das Passwort als userspezifisches Feld anlegen, VAR-Feld (Länge 20, „user_password“ wird zu „pf_user_password“).

```
SELECT u2.pf_user_password AS pass, u2.pf_user_amaccess AS access FROM
%s AS u1, phpbb_profile_fields_data AS u2 WHERE u1.user_id = u2.user_id
AND (u1.username='%s' OR u1.username='%s')
```

Nun kann jeder Nickname oder jede Steam ID in der Datenbank wiedergefunden werden. Eine Steam ID als Nickname ist aber eher unerwünscht. Also wird noch ein weiteres Feld benötigt, das die Steam ID beinhaltet (VARCHAR, Länge 31, „user_steamid“), das vom User im Profil editierbar sein sollte.

```
SELECT u2.pf_user_password AS pass, u2.pf_user_amaccess AS access FROM
%s AS u1, phpbb_profile_fields_data AS u2 WHERE u1.user_id = u2.user_id
AND (u1.username='%s' OR u2.pf_user_steamid='%s')
```

Man kann die Abfrage auch beispielsweise auf bestimmte Gruppen (z.B. Clans) einschränken, so dass ein angemeldeter, normaler User ohne Passwort auf den Server kann bzw. die Server clanweise aus einer Datenbank verteilt werden können. Man könnte das Ganze auch mit den Tags weitertreiben und diese an bestimmte Gruppen binden, etc.

Betreibt man eine Community, könnte man auch auf Passwörter verzichten wollen. Dann muss man aber mühsam, die Abfrage des Usernamens unterbinden:

```
SELECT " AS pass, u2.pf_user_amaccess AS access FROM %s AS u1,
phpbb_profile_fields_data AS u2 WHERE u1.user_id = u2.user_id AND
'%s'='%s' AND u2.user_steamid='%s'
```

Es ist aber auch weiterhin zumindest bei der Übertragung das Passwort in einen MD5-Hash umzuwandeln:

`u2.pf_user_password AS pass` ⟶ `MD5(u2.pf_user_password) AS pass`

Dann muss aber auch `encrypt_password 2` eingestellt werden.

4.12.2. Nutzung von Views

Je komplexer die Abfrage ist, desto schwieriger wird es die von Admin Mod aufgezwunge Syntax einzuhalten. Seit MySQL 5.0.1 ist es möglich sogenannte Views zu erstellen, die die komplexe Abfrage aufnehmen und somit Admin Mod nur noch mit der Standardabfrage genutzt werden muss.

Zunächst das Beispiel für die Einrichtung eines Views mit Passwortabfrage und ausschließlicher Steam ID Nutzung:

```
CREATE VIEW am_users AS SELECT u2.pf_user_steamid AS nick,
u2.pf_user_password AS pass, u2.pf_user_amaccess AS access FROM
phpbb_users AS u1, phpbb_profile_fields_data AS u2 WHERE
```

```
u1.user_id=u2.user_id
```

oder in diesem Fall ohne Passwortabfrage:

```
CREATE VIEW am_users AS SELECT u2.pf_user_steamid AS nick, " AS pass,
u2.pf_user_amaccess AS access FROM phpbb_users AS u1,
phpbb_profile_fields_data AS u2 WHERE u1.user_id=u2.user_id
```

Bitte nicht vergessen nun auf die Tabelle „am_users“ zu verweisen:

```
mysql_dbtable_users "am_users"
```

In den meisten Fällen würde ich Views vorziehen, da man mit Admin Mod nur sehr eingeschränkt auf Fehlersuche gehen kann. Sie sind aber kein Allheilmittel und stehen bei älteren MySQL-Versionen auch nicht zur Verfügung.

4.12.3. Nutzung von phpBB2

Wie bereits eingangs erwähnt, steht jedem die weitere Integration offen. Auch andere CMS oder Foren sind möglich. Es ist ebenfalls die Implementierung in phpBB2 möglich, wenngleich ungleich schwieriger, da zunächst ein Hack wie das Profile Control Center[8] installiert werden muss. Weder dessen Installation noch Konfiguration sind trivial. Auch passt die Installationsanweisung teilweise nicht mehr zur aktuellen Version.

Abschließend die Admin Mod SQL-Abfrage für phpBB2 mit installiertem Profile Control Center und Passwortabfrage sowie den passenden View:

```
SELECT user_password AS pass, user_amaccess AS access FROM %s WHERE
username='%s' OR user_steamid='%s'
```

```
CREATE VIEW am_users AS SELECT user_steamid AS nick, user_password AS
pass, user_amaccess AS access FROM phpbb_users
```

[8] http://www.phpbb.com/community/viewtopic.php?t=150925

5. Standardplugins und Befehle

Bis auf wenige Ausnahmen stammen die Admin Mod Befehle, die man zumeist in der Console eingibt, aus den installierten Plugins.

Bei der Installation von Adminmod werden in der Regel die Standard-Plugins mitinstalliert. In den folgenden Abschnitten soll ein Überblick über die in diesen Plugins zur Verfügung gestellten Befehle gegeben werden. Dabei wird gezeigt, welche Befehle im welchen Plugin stehen und welchen Rechtelevel man für diese braucht.

Parameter von Befehlen, die in eckigen Klammern stehen ([...]), sind optional und müssen nicht angegeben werden. Parameter von Befehlen, die in <...> stehen, sind Pflicht.

5.1. Befehle unabhängig von Plugins

Wenngleich die meisten Admin Mod-Befehle in Plugins definiert werden, gibt es auch einige wenige, die direkt in Admin Mod implementiert sind.

5.1.1. admin_adm

`admin_adm <Option>`

Mit diesem Befehl können einige administrative Befehle, die direkt in Admin Mod verankert sind aufgerufen werden. Derzeit steht nur die Option `mefix` zur Verfügung, mit der Adminmod versucht, verloren gegangene Entities auf einer Map wieder zu reparieren (z.B. nicht sichtbare Leitern, verschwundene Fenster). Dieser Befehl agiert unabhängig von der Adminmod Cvar amv_enable_beta <mefix1>. Genau wie bei der Cvar ist bislang kein positiver Effekt bemerkbar gewesen.

Beispiel:
`admin_cmd admin_adm mefix`

Versucht verloren gegangene Entities zu reparieren.

Access-Level: 0
Gehört zu: kein Plugin

Siehe auch:
amv_enable_beta

5.1.2. admin_cmd

`admin_cmd <Befehl>`

Normalerweise werden die Admin Mod Befehle nur in der Client Console eingeben. Es ist aber auch möglich alle Admin Mod Befehle direkt in der Serverconsole (z.B. via RCon) einzugeben. Dazu muss ein admin_cmd oder ein admin_command dem eigentlichen Befehl vorangestellt werden. Ein solch eingegebener Befehl hat immer ausreichende Rechte.

Beispiel:
`admin_cmd admin_csay Dies ist ein Test!`

Dies schreibt „Dies ist ein Test!“ in die Mitte des Bildschirms aller Spieler auf dem Server (admin_csay).

Access-Level: Serverconsole
Gehört zu: kein Plugin

Siehe auch:
admin_command

5.1.3. admin_command

`admin_command <Befehl>`

Dieser Befehl ist identisch mit admin_cmd, welcher aus Gründen der Kürze empfohlen wird.

Access-Level: Serverconsole
Gehört zu: kein Plugin

Siehe auch:
admin_cmd

5.1.4. admin_help

`admin_help <#/Text>`

Zeigt alle Befehle der Adminmod Plugins alphabetisch in Zehnergruppen an, die man entsprechend seinem Rechtelevel ausführen darf. Man kann entweder eine Zahl (x) angeben, die den x-ten Eintrag und die folgenden 9 Einträge zurückgibt, oder einen Text, bei dem alle Befehle aufgelistet werden, die den Text im Befehl oder im Hilfetext enthalten. Es ist zu beachten, dass nur Befehle angezeigt werden, auf die man auch Zugriff hat.

Beispiele:
`admin_help 11`
`admin_help maps`

Der erste Befehl zeigt die Einträge 11-20 in der Console an. Der zweite zeigt die ersten 10 Einträge, die „maps" im Befehl oder ihrem Hilfetext beinhalten.

Access-Level: 0
Gehört zu: kein Plugin

5.1.5. admin_login

`admin_login <Passwort>`

Mit diesem Befehl kann man sich auf dem Server als Admin anmelden. Es ist lediglich das Passwort aus der users.ini anzugeben. Diese Methode funktioniert nur solange man die Namenreservierung nicht zum Accesslevel hinzugefügt wurde. Anderenfalls wird man gekickt, bevor man den Befehl eintippen kann. Hier muss man auf die automatische Anmeldung zurückgreifen (Kapitel 4.4.2).

Beispiel:
`admin_login 37skcF12p`

Sollte das Passwort „37skcF12p" zu der Spieler-ID oder dem Spielername des Users gehören, weist Admin Mod ihm die entsprechenden Rechte zu.

Access-Level: 0
Gehört zu: kein Plugin

Siehe auch:
admin_password, Namenreservierung, users.ini

5.1.6. admin_password

`admin_password <Passwort>`

Dieser Befehl ist ein Synonym für admin_login. Ursprünglich gab es nur admin_password, was jedoch zu häufig mit admin_pass (Serverpasswort setzen) verwechselt wurde. Somit wurde admin_login als Ersatz eingeführt. Der Befehl admin_password existiert nur zur Abwärtskompatibilität.

Beispiel:
`admin_password 37skcF12p`

Sollte das Passwort „37skcF12p" zu der Spieler-ID oder dem Spielername des Users gehören, weist Admin Mod ihm die entsprechenden Rechte zu.

Access-Level: 0
Gehört zu: kein Plugin

Siehe auch:
admin_login, Namenreservierung, users.ini

5.1.7. admin_status

`admin_status [ID/Name]`

Mit admin_status kann der Accesslevel erfragt werden, der einem von Admin Mod zugewiesen wurde. Genauso kann man auch erfragen, welchen Accesslevel ein anderer Spieler hat.

Beispiele:
`admin_status`

`admin_status Player1`

Im ersten Beispiel wird der eigene Accesslevel in der Console angezeigt, während im zweiten Beispiel der Accesslevel des Spielers „Player1" angezeigt wird.

Access-Level: 0
Gehört zu: kein Plugin

Siehe auch:
users.ini

5.1.8. admin_version

`admin_version`

Wie der Name bereits erahnen lässt, wird die Versionsnummer der geladenen Admin Mod Version ausgegeben. Darüber hinaus werden aber auch alle installierten Admin Mod Plugins ausgegeben.

Beispiele:
`admin_version`

Gibt die Admin Mod Version und alle installierten Plugins aus.

Access-Level: 0
Gehört zu: kein Plugin

Siehe auch:
plugin.ini

5.2. plugin_antiflood

Das Antiflood Plugin beinhaltet keine Befehle. Es soll den Server vor Leuten schützen, die durch das Absetzen unzähliger Befehle und Chatnachrichten versuchen den Server zum Absturz zu bringen. Es ist daher wichtig, dass das Plugin in der plugin.ini immer an der ersten Stelle steht (wird dann als erstes geladen und bei einer Anfrage ausgeführt).

5.3. plugin_base

Das Base Plugin beinhaltet alle grundlegenden Befehle, die für die Administration mehr oder weniger wichtig sind (insbesondere das Setzen von Servervariablen). Die zur Verfügung stehenden Befehle beziehen sich auf die Basisfunktionen des HL-Servers und sind damit unabhängig von der eingesetzten Modifikation.

Als weitere Option bietet das Plugin an, mapspezifische Einstellungen aus einer Konfigurationsdatei zu laden. Man muss lediglich eine Datei mit dem Namen der Map ins Modverzeichnis erstellen (z.B. de_dust.cfg) und die Sondereinstellungen hinterlegen. In diesem Fall bitte sicherstellen, dass mapchangecfgfile in der server.cfg auf „server.cfg“ gesetzt wird (ggf. die Variable ergänzen). Anderenfalls werden bei einer anderen Map die Einstellungen nicht mehr rückgängig gemacht.

5.3.1. admin_abort_vote

`admin_abort_vote`

Der Befehl bricht einen laufenden Mapvote (admin_vote_map) oder aber auch einen Kickvote (admin_vote_kick) ab.

Beispiele:
`admin_abort_vote`

Bricht den laufenden Vote ab, oder gibt zurück, dass kein Vote läuft.

Access-Level: 2
Gehört zu: plugin_base

Siehe auch:
admin_vote_kick, admin_vote_map

5.3.2. admin_ban

`admin_ban <ID/Namensteil/IP> [Minuten] [IP]`

Mit diesem Befehl wird der Spieler entweder über seine ID, IP oder den Namen für eine bestimmte Anzahl an Minuten gebannt. Sofern keine IP angegeben wurde, kann man zudem angeben, ob der Spieler statt mit seiner ID mit seiner IP gebannt wird. Die Minutenzahl ist optional. Standardmäßig wird 0 angenommen, was ein permanenter Ban ist. Im Regelfall (`sv_lan 0`) wird auch angenommen, dass eine ID gebannt werden soll. Erst wenn man es explizit am Ende des Befehls angibt oder bei `sv_lan 1`, wird die IP gebannt.

Beim Bannen einer ID oder IP muss der Spieler nicht auf dem Server sein.

Bei einem LAN-Server ohne Internetzugang kann auch gleich admin_banip genutzt werden.

Beispiele (sv_lan 0):

```
admin_ban Idiot
admin_ban STEAM_0:1:234567 60
admin_ban STEAM_0:1:234567 60 IP
admin_ban 81.56.23.12 60
```

Im ersten Beispiel wird ein Spieler mit dem Namen „Idiot“ permanent mit seiner Steam ID gebannt. Es kann sich auch um einen Teil des Namens handeln, z.B. „Ich bin ein Idiot“. Solange der Name mit „Idiot“ eindeutig zuzuordnen ist, wird der Spieler gebannt. Sind beispielsweise zwei Spieler „Ich bin ein Idiot“ und „Ich war ein Idiot“ kommt es zu keinem Bann, da Admin Mod den Bann nicht eindeutig zuordnen kann.

Im zweiten Beispiel wird der Spieler mit der Steam ID „STEAM_0:1:234567“ für 60 Minuten über seine Steam ID gebannt. Im dritten Beispiel hingegen wird nicht seine Steam ID sondern seine IP gebannt.

Das letzte Beispiel basiert auf der IP des Spielers. Allerdings wird nicht seine IP sondern seine Steam ID für 60 Minuten gebannt.

Access-Level: 256
Gehört zu: plugin_base

Siehe auch:
admin_banip, admin_unban

5.3.3. admin_banip

`admin_banip <ID/Namensteil/IP> [Minuten]`

Mit diesem Befehl wird der Spieler entweder über seine ID, IP oder den Namen für eine bestimmte Anzahl an Minuten gebannt. Allerdings wird nur seine IP gebannt. Die Minutenzahl ist optional. Standardmäßig wird 0 angenommen, was ein permanenter Ban ist.

Der Befehl admin_ban deckt die Funktionalität von admin_banip ab. Man ist dort aber dann gezwungen eine Minutenzahl anzugeben, um eine IP zu bannen.

Beim Bannen einer IP muss der Spieler nicht auf dem Server sein.

Beispiele:

```
admin_banip Idiot
admin_banip STEAM_0:1:234567 60
admin_ban 81.56.23.12 60
```

Im ersten Beispiel wird ein Spieler mit dem Namen „Idiot“ permanent mit seiner IP gebannt. Es kann sich auch um einen Teil des Namens handeln, z.B. „Ich bin ein Idiot“. Solange der Name mit „Idiot“ eindeutig zuzuordnen ist, wird der Spieler gebannt. Sind beispielsweise zwei Spieler „Ich bin ein Idiot“ und „Ich war ein Idiot“ kommt es zu keinem Bann, da Admin Mod den Bann nicht eindeutig zuordnen kann.

Im zweiten Beispiel wird der Spieler mit der Steam ID „STEAM_0:1:234567“ für 60 Minuten über seine IP gebannt.

Das letzte Beispiel basiert auf der IP des Spielers.

Access-Level: 256
Gehört zu: plugin_base

Siehe auch:
admin_ban, admin_unban

5.3.4. admin_cfg

`admin_cfg <Dateiname>`

Der Befehl ermöglicht das Ausführen der angegebenen Konfigurationsdatei auf dem Server.

Beispiel:
`admin_cfg server.cfg`

In diesem Beispiel wird die server.cfg ausgeführt, so dass alle Einstellungen aus dieser aktiv werden.

Access-Level: 512
Gehört zu: plugin_base

Siehe auch:
admin_cfg

5.3.5. admin_chat

`admin_chat <Text>`

Mittels dieses Befehls kann ein Admin allen anderen Admins gleichzeitig Nachrichten übermitteln, ohne dass ein normaler Spieler dies mitbekommt. Berechtigte Admins zum Absenden und Empfangen von Nachrichten sind alle Admins, die das Recht 64 besitzen.

Beispiel:
`admin_chat Der Spieler CatchMe ist vermutlich ein Cheater.`

Hier meldet ein Admin den anderen, dass er einen Spieler „CatchMe“ als Cheater in Verdacht hat, ohne dass die anderen normalen Spieler dies mitbekommen.

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_csay, admin_dmesg, admin_psay, admin_say, admin_ssay, admin_tsay, admin_vsay

5.3.6. admin_csay

`admin_csay [Farbe] <Text>`

Dieser Befehl schreibt den gewünschten Text mittig in den Bildschirm aller Spieler. Standardmäßig (ohne Farbangabe) wird es in grün dargestellt. Mögliche Farben sind „red“, „blue“, „green“, „white“, “yellow“, „purple“ und „random“ (Zufall). Einen Zeilenumbruch bekommt man durch Eingabe von „\n“.

Beispiel:
`admin_csay red Vorsicht! Bissiger Admin.`

Der Admin deutet in leuchtenden Lettern an, dass mit ihm nicht zu spaßen ist.

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_dmesg, admin_psay, admin_say, admin_ssay, admin_tsay, admin_vsay

5.3.7. admin_dmesg

`admin_dmesg <ID-Typ> <ID> <Text>`

Dieser Befehl ist etwas ungewöhnlich, da der angeredete Spieler nicht über seinen Namen sondern einer seiner IDs adressiert. Beim ID-Typ stehen 3 Optionen zur Auswahl. Bei „i“ wird der Userindex (Slotnummer), bei „s“ seine Session ID und bei „w“ seine WONID/Steam ID erwartet.

Bequemer ist in jedem Fall admin_psay, da außer der Steam ID die IDs meist unbekannt sind.

In der Vor-Steam Zeit konnte man damit Nachrichten an einen Spieler senden, der gerade dem Server beitrat.

Beispiele:

```
admin_dmesg i 3 Benimm Dich!!
admin_dmesg s 449 Benimm Dich!!
admin_dmesg w STEAM_0:1:234567 Benimm Dich!!
```

In allen drei Fällen wird der Spieler (mit dem Userindex 3, der Session ID 449 oder der Steam ID STEAM_0:1:234567) auf sein schlechtes Verhalten hingewiesen.

Access-Level: 512
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_psay, admin_say, admin_ssay, admin_tsay, admin_vsay

5.3.8. admin_fraglimit

`admin_fraglimit <Wert>`

Mit diesem Eintrag kann man den Wert der Servervariablen „mp_fraglimit“ verändern.

Exkurs: Diese Variable sorgt für einen Mapwechsel, wenn der beste Spieler die angegebene Frag-Zahl (Kills) erreicht hat. 0 schaltet die Funktion ab.

Beispiele:

```
admin_fraglimit 50
admin_fraglimit 0
```

Im ersten Beispiel wird das Frag-Limit auf 50 Frags gesetzt, während im zweiten Beispiel die Funktion abgeschaltet wird.

Access-Level: 2
Gehört zu: plugin_base

5.3.9. admin_friendlyfire

`admin_friendlyfire <Wert>`

Mit diesem Befehl kann man den Wert der Servervariablen „mp_friendlyfire“ verändern.

Exkurs: Diese Variable stellt ein, ob man vom eigenen Team getroffen werden kann (Friendly Fire) oder nicht. „1“ schaltet die Friendly Fire ein, „0“ schaltet es aus.

Beispiele:

```
admin_friendlyfire 1
admin_friendlyfire 0
```

Im ersten Beispiel wird Friendly Fire eingeschaltet, während im zweiten Beispiel die Funktion abgeschaltet wird.

Access-Level: 32
Gehört zu: plugin_base

5.3.10. admin_gravity

`admin_gravity <Wert>`

Diesem Befehl erlaubt die Schwerkraft durch das Setzen der Servervariablen „sv_gravity“ zu verändern.

Exkurs: Diese Variable stellt ein, wie hoch die Schwerkraft auf dem Server ist. Überlicherweise steht diese auf „800“.

Beispiele:
`admin_gravity 300`
`admin_gravity 800`

Das erste Beispiel erlaubt weite Sprünge, während das zweite die Schwerkraft wieder auf normale Werte zurücksetzt.

Access-Level: 32
Gehört zu: plugin_base

5.3.11. admin_hostname

`admin_hostname <Wert>`

Diesem Befehl erlaubt den Namen des Gameservers zu verändern. Dabei wird die Servervariable „hostname“ geändert.

Exkurs: Die Variable „hostname“ bestimmt, wie der Server heißt, und wie er beispielsweise im Steam-Browser dargestellt wird.

Beispiele:
`admin_hostname Mein toller HL-Server`

In diesem Beispiel ändert man den Servernamen in „Mein toller HL-Server“.

Access-Level: 512
Gehört zu: plugin_base

5.3.12. admin_kick

`admin_kick <ID/Namensteil/IP> [Grund]`

Mit diesem Befehl wird der Spieler entweder über seine ID, IP oder den Namen vom Server geworfen (Kick). Darüber hinaus kann dem Spieler auch ein Grund für den Kick gegeben werden. Er sieht dies nach dem Muster „You have been kicked because ...“. Statt der „...“ kann man den Grund eintragen.

Beispiele:

```
admin_kick Idiot
admin_kick "Ich bin ein Idiot" "you are afk."
```

Im ersten Beispiel wird ein Spieler mit dem Namen „Idiot“ ohne Gründe zu nennen vom Server geworfen. Es kann sich auch um einen Teil des Namens handeln, z.B. „Ich bin ein Idiot“. Solange der Name mit „Idiot“ eindeutig zuzuordnen ist, wird der Spieler gekickt. Sind beispielsweise zwei Spieler „Ich bin ein Idiot“ und „Ich war ein Idiot“ kommt es zu keinem Kick, da Admin Mod den Kick nicht eindeutig zuordnen kann.

Im zweiten Beispiel wird der Spieler mit dem Namen „Ich bin ein Idiot“ vom Server geworfen, weil er nicht am Rechner sitzt und mitspielt (afk = away from keyboard). Der Spieler bekommt dann die Meldung „You have been kicked because you are afk“. Man beachte auch die Anführungszeichen. Gibt es Leerzeichen im Spielernamen oder in der Nachricht, muss der Text jeweils in Anführungszeichen gesetzt werden.

Access-Level: 128
Gehört zu: plugin_base

Siehe auch:
admin_vote_kick

5.3.13. admin_listmaps

`admin_listmaps`

Gibt die Maps im Mapcycle und die derzeitige Map in der Console aus.

Beispiele:
`admin_listmaps`

Dies führt beispielsweise zu folgendem Ausdruck in der Console:

```
The maps on the mapcycle are:
de_dust
de_dust2
de_aztec
and the current map is:
de_dust2
```

Access-Level: 0
Gehört zu: plugin_base

5.3.14. admin_map

`admin_map <Map>`

Dieser Befehl führt einen sofortigen Mapwechsel zur angegebenen Map durch.

Beispiel:
`admin_map de_dust2`

In diesem Beispiel wird mit sofortiger Wirkung zur Map „de_dust2“ gewechselt.

Access-Level: 2
Gehört zu: plugin_base

Siehe auch:
admin_vote_map

5.3.15. admin_motd

```
admin_motd <Spieler> <Text>
```

Dieser Befehl öffnet beim angegebenen Spieler das MotD (Message of the Day) Fenster, das den gewünschten Text beinhaltet. Dieser Befehl funktioniert nicht bei allen Mods, da nur einige das MotD Fenster unterstützen (z.B. Counter-Strike, aber nicht TFC oder Ricochet). Zudem ist die Länge des Textes sehr eingeschränkt.

Beispiel:

```
admin_motd SucheClan <html><head><meta http-equiv="REFRESH" content="0;
url=www.google.de"></head><body></body></html>
```

In diesem Beispiel verweisen wir den Spieler „SucheClan“ durch einen HTML-Refresh auf Google, damit er dort weitersucht...

Access-Level: 64
Gehört zu: plugin_base

5.3.16. admin_nextmap

```
admin_nextmap
```

Mit diesem Befehl kann man nachsehen, was die nächste Map sein wird. Dies setzt voraus, dass durch eine Mapvote nicht eine andere Map gewählt wurde. Außerdem gibt es eine Einschränkung beim Start des Servers. Erst, wenn der erste reguläre Mapwechsel (Half-Life wechselt die Map, nicht Admin Mod) stattgefunden hat, wird die nächste Map korrekt angezeigt. Alternativ sollte man die letzte Map im Mapcycle gleich der Startmap setzen. Dann ist auch dieses Problem gelöst.

Beispiel:

```
admin_nextmap
```

Zeigt die nächste Map an, die nach dem nächsten regulären Mapwechsel kommen sollte.

Access-Level: 0
Gehört zu: plugin_base

Siehe auch:
say nextmap

5.3.17. admin_nopass

`admin_nopass`

Dieser Befehl entfernt das Serverpasswort. Er macht das Gleiche wie `admin_pass ""`. Exkurs: Genau genommen, wird die Servervariable sv_password einfach geleert.

Beispiel:
`admin_nopass`

Entfernt das Serverpasswort.

Access-Level: 16
Gehört zu: plugin_base

Siehe auch:
admin_pass

5.3.18. admin_pass

`admin_pass [Passwort]`

Der Befehl admin_pass kann sowohl das Serverpasswort setzen, entfernen als auch anzeigen. Zum Entfernen des Passworts kann auch `admin_nopass` verwendet werden.

Beispiele:

```
admin_pass Geheim
admin_pass "Geheimes Passwort"
admin_pass ""
admin_pass none
admin_pass
```

Das erste und zweite Beispiel führt zu einem Setzen bzw. Ändern des Serverpassworts. Hat das Serverpasswort ein Leerzeichen, muss es wie im zweiten Beispiel in Anführungszeichen gesetzt werden.

Im dritten und vierten Beispiel wird das Passwort entfernt.

Gibt man, wie im fünften Beispiel gezeigt, nichts weiter an, wird dem Admin lediglich das aktuelle Serverpasswort angezeigt an.

Access-Level: 16
Gehört zu: plugin_base

Siehe auch:
admin_nopass

5.3.19. admin_pause

`admin_pause`

Dieser Befehl ist nicht mit dem Pausieren des Servers zu verwechseln. Es wird lediglich den Spielern erlaubt das Spiel über den Clientbefehl „pause“ anzuhalten. Exkurs: Es wird die Servervariable „pausable“ auf „1“ gesetzt.

Zurücksetzen kann man dies mit dem Befehl admin_unpause.

Beispiel:
`admin_pause`

Erlaubt ALLEN Spielern das Pausieren des Spiels.

Access-Level: 8
Gehört zu: plugin_base

Siehe auch:
admin_unpause

5.3.20. admin_psay

`admin_psay <ID/Namensteil/IP> <Nachricht>`

Der Befehl admin_psay ermöglicht es, persönliche Nachrichten an einen bestimmten Spieler zu schicken. Dabei kann die ID, die IP oder der Name als Empfänger gewählt werden. Es kann sich auch um einen Teil des Namens handeln, z.B. „Ich bin ein Idiot“. Solange der Name mit „Idiot“ eindeutig zuzuordnen ist, wird die Nachricht an den Spieler übermittelt. Sind beispielsweise zwei Spieler „Ich bin ein Idiot“ und „Ich war ein Idiot“ wird die Nachricht nicht übermittelt, da Admin Mod den Empfänger nicht eindeutig zuordnen kann.

Ein Besonderheit stellt der Empfänger „admin“ dar. Durch diesen kann man sich mit einem Admin unterhalten, der sich gerade die Logs auf den Rechner zieht (z.B. mit HLSW[1]). Dieser kann im übrigen auch mit admin_psay über RCon antworten.

Ähnlich funktioniert admin_dmesg, es ist aber deutlich schwieriger zu bedienen.

Beispiele:
```
admin_psay Kleinkind Benimm Dich!!
admin_psay 81.56.23.12 Benimm Dich!!
admin_psay STEAM_0:1:234567 Benimm Dich!!
admin_psay admin Ich bin nicht würdig, ich bin Staub!!
```

[1] http://www.hlsw.de/

Im ersten Beispiel wird der Spieler „Kleinkind“ oder derjenige, dessen Spielername „Kleinkind“ beinhaltet, angehalten, sein Verhalten zu überdenken. Das Gleiche wird für die IP bzw. Steam ID im Beispiel zwei und drei gemacht.

Das letzte Beispiel zeigt eine Würdigung der Arbeit des Admins, der sich die Logs an ein Tool auf seinem Rechner übertragen lässt.

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_dmesg, admin_say, admin_ssay, admin_tsay, admin_vsay

5.3.21. admin_rcon

`admin_rcon <Befehl>`

Mit admin_rcon können beliebige Serverbefehle ausgeführt bzw. Servervariablen gesetzt werden. Ausgenommen davon ist das Ändern des RCon-Passworts (rcon_password), damit Admins, die das Passwort nicht kennen, aber Zugriff auf diesen Befehl haben, sich auf diesem Weg direkten RCon-Zugriff verschaffen.

Es soll an dieser Stelle betont werden, dass dieser Befehl nur in die Hände sehr ausgewählter Admins gegeben werden sollte. In der Regel stellt Admin Mod schon ausreichend Befehle zur Verfügung, so dass man auf admin_rcon verzichten kann.

Beispiele:
```
admin_rcon changelevel de_dust
admin_rcon sv_gravity 600
```

Im ersten Beispiel wird ein Mapwechsel auf de_dust durchgeführt. Dies entspricht dem Admin Mod Befehl `admin_map de_dust`.

Beim zweiten Beispiel wird die Schwerkraft auf 600 gesetzt, was dem Admin Mod Befehl `admin_gravity 600` entspricht.

Access-Level: 65536
Gehört zu: plugin_base

5.3.22. admin_reload

`admin_reload`

Änderungen an den Admin Mod Konfigurationsdateien werden normalerweise nur beim Serverstart bzw. beim Mapwechsel neu eingelesen. Während die adminmod.cfg noch mit `admin_rcon exec addons/adminmod/config/adminmod.cfg` neu geladen werden kann, erfordern die anderen Dateien eine andere Behandlung. Durch das Ausführen von admin_reload werden diese Dateien neu eingelesen (z.B. auch die users.ini).

Besonders wichtig ist der Befehl, wenn die vault.ini manuell editiert wurde. Sollte ein Plugin zwischenzeitlich etwas in die vault.ini, sind alle Änderungen verloren gegangen. Admin Mod liest alle Einträge in einem Rutsch, schreibt sie aber auch in einem Rutsch, so dass auch alle anderen Einträge überschrieben werden. Daher sollte nach dem Editieren schnellstmöglich admin_reload ausgeführt werden

Die plugin.ini kann damit nicht neu eingelesen werden. Eine Änderung an der plugin.ini erfordert mindestens einen Mapwechsel, da nur dann die Plugins neu geladen werden.

Beispiel:
`admin_reload`

Liest die Konfigurationsdateien (außer der adminmod.cfg und plugin.ini) neu ein.

Access-Level: 4
Gehört zu: plugin_base

5.3.23. admin_say

`admin_say <Nachricht>`

Mit diesem Befehl kann eine Nachricht an alle Spieler auf dem Server geschrieben werden. Es wird explizit vermerkt, dass die Nachricht von einem Admin kommt. Im Gegensatz zu admin_ssay wird der Name des Admins angegeben.

Beispiel:
`admin_say Dieser Server geht offline!`

Dies weist die Spieler im Chat darauf hin, dass der Server gleich nicht mehr zur Verfügung steht. Die Meldung sähe z.B. folgendermaßen aus:

`Message from Admin ([WING] Black Knight): Dieser Server geht offline!`

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_dmesg, admin_psay, admin_ssay, admin_tsay, admin_vsay

5.3.24. admin_servercfg

`admin_servercfg <Dateiname>`

Dieser Befehl ermöglicht es, die Serverdatei zu definieren, die beim Serverstart geladen werden soll (Standard: server.cfg). Die Servervariable lautet `servercfgfile`.

Er funktioniert zwar noch, ist aber inzwischen relativ nutzlos geworden. Die geänderte Einstellung wird nicht in die autoexec.cfg geschrieben und steht somit beim Serverstart nicht zur Verfügung. Vormals wurde die angegebene Datei auch immer beim Mapwechsel geladen. Inzwischen wird dies über die Variable `mapchangecfgfile` festgelegt.

Beispiel:
`admin_servercfg server2.cfg`

Statt der server.cfg wird standardmäßig die server2.cfg geladen, was aber nicht mehr relevant ist, da die Einstellung nur beim Serverstart genutzt wird.

Access-Level: 512
Gehört zu: plugin_base

5.3.25. admin_ssay

`admin_ssay <Nachricht>`

Mit diesem Befehl kann eine Nachricht an alle Spieler auf dem Server geschrieben werden. Es wird explizit vermerkt, dass die Nachricht von einem Admin kommt. Im Gegensatz zu admin_say wird der Name des Admins nicht angegeben.

Beispiel:
`admin_ssay Dieser Server geht offline!`

Dies weist die Spieler im Chat darauf hin, dass der Server gleich nicht mehr zur Verfügung steht. Die Meldung sähe z.B. folgendermaßen aus:

```
Dieser Server geht offline!
```

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_dmesg, admin_psay, admin_say, admin_tsay, admin_vsay

5.3.26. admin_teamplay

`admin_teamplay <0/1>`

Hiermit kann die Servervariable mp_teamplay verändert werden, die bei bestimmten Mods den Wechsel zwischen Teamplay und Deathmatch erlaubt.

Beispiel:
`admin_teamplay 1`

Der Server wechselt in den Team-Modus.

Access-Level: 32
Gehört zu: plugin_base

5.3.27. admin_timeleft

`admin_timeleft`

Dieser Befehl erlaubt die Angabe der Restzeit bis zum Mapwechsel im Chat.

Das funktioniert allerdings nicht bei allen Modifikationen zuverlässig. Gerade bei Counter-Strike zählt nur der Client die Restzeit nicht aber der Server. Die Rückgabe der Restzeit in Counter-Strike ist damit ungenau. Einige Custom-Plugins nehmen sich dieses Umstands an.

Es gibt auch die Möglichkeit den Chat zu verwenden (say timeleft)

Beispiel:
`admin_timeleft`

Gibt die Restzeit bis zum Mapwechsel im Chat an.

Access-Level: 0
Gehört zu: plugin_base

Siehe auch:
say timeleft

5.3.28. admin_timelimit

`admin_timelimit <Minuten>`

Hiermit kann die Servervariable mp_timelimit verändert werden. Diese legt fest, wie lang eine Map läuft. Die Angabe wird in Minuten erwartet (0 deaktiviert den Mapwechsel durch das Ablaufen eines Countdowns).

Beispiel:
`admin_timelimit 45`

Die Laufzeit einer Map wird auf 45 Minuten gesetzt.

Access-Level: 2
Gehört zu: plugin_base

5.3.29. admin_tsay

`admin_tsay [Farbe] <Text>`

Dieser Befehl schreibt den gewünschten Text auf der linken Seite des Bildschirms aller Spieler. Standardmäßig (ohne Farbangabe) wird es in hellgrau dargestellt. Mögliche Farben sind „red“, „blue“, „green“, „white“, “yellow“, „purple“ und „random“ (Zufall). Einen Zeilenumbruch bekommt man durch Eingabe von „\n“.

Beispiel:
`admin_tsay red Vorsicht! Bissiger Admin.`

Der Admin deutet in leuchtenden Lettern an, dass mit ihm nicht zu spaßen ist.

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_dmesg, admin_psay, admin_say, admin_ssay, admin_vsay

5.3.30. admin_unban

`admin_unban <ID/IP>`

Mit diesem Befehl wird der Ban gegen eine bestimmte Spieler ID oder IP aufgehoben. Selbstverständlich muss der Spieler dafür nicht auf dem Server sein.

Beispiele:
`admin_unban STEAM_0:1:234567`
`admin_unban 81.56.23.12`

Im ersten Beispiel wird die Steam ID „STEAM_0:1:234567“ entbannt, während im zweiten Beispiel die IP „81.56.23.12“ entbannt wird.

Access-Level: 256
Gehört zu: plugin_base

Siehe auch:
admin_ban, admin_banip

5.3.31. admin_unpause

`admin_unpause`

Dieser Befehl ist nicht mit dem Aufheben der Serverpausierung zu verwechseln. Es wird lediglich den Spielern nicht mehr erlaubt das Spiel über den Clientbefehl „pause“ anzuhalten. Exkurs: Es wird die Servervariable „pausable“ auf „0“ gesetzt.

Zurücksetzen kann man dies mit dem Befehl admin_pause.

Beispiel:
`admin_unpause`

Verbietet ALLEN Spielern das Pausieren des Spiels.

Access-Level: 8
Gehört zu: plugin_base

Siehe auch:
admin_pause

5.3.32. admin_userlist

`admin_userlist [Muster]`

Dieser Befehl zeigt ein Liste aller Spieler auf dem Server in der Console an. Wird ein Muster angebenen, werden nur diejenigen Spieler angezeigt, deren Name mit diesem Muster beginnt.

Als weitere Informationen werden neben dem Namen die Session-ID, die Steam ID, der Accesslevel und die Immunität angezeigt.

Beispiele:
```
admin_userlist
admin_userlist N
```

Im ersten Beispiel werden alle Spieler auf dem Server in der Console aufgelistet.

Im zweiten Beispiel hingegen werden nur die Spieler gezeigt, die mit „N“ im Namen beginnen.

Access-Level: 0
Gehört zu: plugin_base

5.3.33. admin_vote_kick

`admin_vote_kick <ID/Namensteil/IP>`

Mit diesem Befehl wird ein Vote ausgelöst, bei dem die Mehrheit der Spieler entscheidet, ob der benannte Spieler vom Server geworfen wird oder nicht. Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Beispiele:
```
admin_vote_kick Idiot
admin_vote_kick "Ich bin ein Idiot"
```

Im ersten Beispiel wird ein Vote zum Spieler mit dem Namen „Idiot“ durchgeführt, ob dieser vom Server geworfen wird. Es kann sich auch um einen Teil des Namens handeln, z.B. „Ich bin ein Idiot“. Solange der Name mit „Idiot“ eindeutig zuzuordnen ist, wird Vote ausgeführt. Sind beispielsweise zwei Spieler „Ich bin ein Idiot“ und „Ich war ein Idiot“ kommt es nicht zum Vote, da Admin Mod den Vote nicht eindeutig zuordnen kann.

Im zweiten Beispiel wird ein Vote zum Spieler mit dem Namen „Ich bin ein Idiot“ ausgeführt. Gibt es Leerzeichen im Spielernamen, muss der Text jeweils in Anführungszeichen gesetzt werden.

Weiterhin ist zu beachten, dass Votes nicht beliebig kurz hintereinander durchgeführt werden können (vote_freq).

Ohne Votum der Spieler kann man auch jemanden über admin_kick vom Server werfen.

Access-Level: 1
Gehört zu: plugin_base

Siehe auch:
admin_kick, vote_freq

5.3.34. admin_vote_map

`admin_vote_map <Map>`

Dieser Befehl führt ein Vote aus, ob zur angegebenen Map gewechselt werden soll. Sollte sich die Mehrheit für einen Wechsel entscheiden, wird dieser durchgeführt.

Beispiel:
`admin_vote_map de_dust2`

In diesem Beispiel wird ein Vote zur Map „de_dust2“ durchgeführt. Es ist bei dieser Map äußerst wahrscheinlich, dass sich eine Mehrheit der Spieler für „de_dust2“ entscheidet und der Mapwechsel vollzogen wird.

Weiterhin ist zu beachten, dass Votes nicht beliebig kurz hintereinander durchgeführt werden können (vote_freq).

Ohne Votum der Spieler kann man auch die Map mit admin_map wechseln.

Access-Level: 1
Gehört zu: plugin_base

Siehe auch:
admin_map, vote_freq

5.3.35. admin_vsay

`admin_vsay <Frage>`

Mit diesem Befehl kann man eine beliebige Frage, die mit „Ja“ oder „Nein“ beantwortet werden kann, an die Spieler stellen. Das Ergebnis wird im Chat dargestellt, hat aber keine automatische Aktion zur Folge. Einen Zeilenumbruch in der Frage bekommt man durch Eingabe von „\n“.

Beispiel:
`admin_vsay Seid Ihr fertig?`

Zum Start eine Clanwars könnte man fragen, ob alle Beteiligten fertig sind. Wählen alle Spieler „Ja“ aus, kann der Admin den Clanwar starten.

Access-Level: 64
Gehört zu: plugin_base

Siehe auch:
admin_chat, admin_csay, admin_dmesg, admin_psay, admin_say, admin_ssay, admin_tsay, vote_freq

5.4. plugin_chat

Das Chat Plugin beinhaltet nicht sehr viele Funktionen. Allen Befehlen ist aber gemeinsam, dass sie nicht in die Console sondern im Chat eingegeben werden oder die Eingabe von Befehlsoptionen im Chat erlauben.

5.4.1. admin_messagemode

`admin_messagemode <Befehl>`

Dieser Befehl erlaubt es, die Optionen eines Befehls im Chat einzugeben. Dabei ist es unerheblich, ob es sich um einen originären Clientbefehl oder einen Admin Mod Befehl handelt. Will man einen originären Serverbefehl absetzen, so ist „rcon“ dem Befehl voran zu stellen. Das richtige RCon-Passwort muss daher beim ausführenden Spieler gesetzt sein. Dies verhindert, dass Spieler über admin_messagemode RCon-Rechte erhalten können. Admin Mod Befehle überprüfen die Rechte des Spielers automatisch.

Zunächst mag es seltsam erscheinen, warum nur Optionen im Chat eingegeben werden können. Die dahinterliegende Idee basiert jedoch darauf, dass man sich Bindings auf Tasten legen kann. Dadurch muss man z.B. nicht in die Console wechseln, wenn man beispielsweise jemanden kicken möchte.

Man sollte aber aufpassen. Sobald man admin_messagemode aktiviert, werden alle folgenden Chat-Eingaben als Befehlsoptionen aufgefasst. Das kann dann beispielsweise zu Fehlaktionen führen. Nach durchgeführter Aktion sollte daher die Funktion wieder abgeschaltet werden. Dies kann durch das Leerstehen des Befehls, durch „say“ als Befehl oder durch admin_nomessagemode erreicht werden.

Beispiele:

```
admin_messagemode admin_kick
admin_messagemode rcon
admin_messagemode buy
admin_messagemode
admin_messagemode say
```

Im ersten Beispiel wird der Befehl admin_kick eingespeichert. Jede Chatnachricht wird nun als Spielername interpretiert bis der Messagemode wieder aufgehoben wird.

Im zweiten Beispiel können Serverbefehle im Chat abgesetzt werden, sofern der Spieler das RCon-Passwort bereits eingegeben hat.

Das dritte Beispiel zeigt wie man im Chat einkaufen kann. „hegren“ in den Chat eingegeben kauft in Counter-Strike beispielsweise eine Handgranate.

Die letzten beiden Beispiele schalten den Messagemode ab.

Wie bereits erwähnt lohnt sich der Messagemode nur, wenn man ihn in Bindings nutzt.

Access-Level: 0
Gehört zu: plugin_chat

Siehe auch:
admin_nomessagemode

5.4.2. admin_nomessagemode

`admin_nomessagemode`

Dieser Befehl schaltet den aktivierten Messagemode (admin_messagemode) ab.

Beispiel:
`admin_nomessagemode`

Dies schaltet den Messagemode ab, nicht mehr und nicht weniger.

Access-Level: 0
Gehört zu: plugin_chat

Siehe auch:
admin_messagemode

5.4.3. say currentmap

`say currentmap`

Gibt man „currentmap“ im Chat ein, wird der Name der Map angezeigt, auf der gerade gespielt wird.

Beispiel:
`say currentmap`

Siehe Beschreibung.

Access-Level: 0
Gehört zu: plugin_chat

Siehe auch:
say nextmap

5.4.4. say nextmap

`say nextmap`

Gibt man „nextmap“ im Chat ein, wird der Name der Map angezeigt, die als nächstes folgenden würde, wenn nicht ein Eingriff seitens eines Server-Addons/Plugins oder des Admins erfolgt.

Beim Serverstart ist die Anzeige auf der ersten Map inkorrekt. Alternativ sollte man die letzte Map im Mapcycle gleich der Startmap setzen. Dann ist auch diese Anzeige korrekt.

Alternativ kann auch der Befehl admin_nextmap verwendet werden.

Beispiel:
`say nextmap`

Zeigt die nächste Map an.

Access-Level: 0
Gehört zu: plugin_chat

Siehe auch:
admin_nextmap, say nextmap

5.4.5. say timeleft

`say timeleft`

Gibt man „timeleft“ im Chat ein, wird die verbleibende Zeit angezeigt, die auf der entsprechenden Map noch gespielt wird.

Das funktioniert allerdings nicht bei allen Modifikationen zuverlässig. Gerade bei Counter-Strike zählt nur der Client die Restzeit nicht aber der Server. Die Rückgabe der Restzeit in Counter-Strike ist damit ungenau. Einige Custom-Plugins nehmen sich dieses Umstands an.

Alternativ kann auch der Befehl admin_timeleft verwendet werden.

Beispiel:
`say timeleft`

Zeigt die verbleibende Zeit auf der derzeitigen Map an.

Access-Level: 0
Gehört zu: plugin_chat

Siehe auch:
admin_timeleft

5.5. plugin_cheat

Das plugin_cheat erweitert die Fähigkeiten des Admins auf ein Maß, das beim Spielen als Cheaten bezeichnet werden muss. Die Befehle haben meist zwar auch ganz sinnvolle Funktionen, die aber natürlich auch in „böser“ Absicht verwendet werden können. Bei den besonders ins Spiel eingreifenden Befehlen, gibt es eine Meldung an alle Spieler auf dem Server. Diese wird unabhängig von der Einstellung admin_quiet abgesetzt.

Letztendlich wurde bereits darüber nachgedacht, die Funktionen hinter admin_godmode und admin_noclip zu entfernen. Bis dato sind sie aber noch vorhanden.

5.5.1. admin_godmode

`admin_godmode <Spielername> <"on" oder "off">`

Der Befehl admin_godmode erlaubt dem Admin, einem Spieler die Fähigkeit zu geben, keinen Schaden zu nehmen. Eine serverweite Meldung wird automatisch abgesetzt, sobald der Befehl ausgeführt wird! Diese wird unabhängig von der Einstellung admin_quiet abgesetzt.

Gedacht war der Befehl ursprünglich, beim Deathmatch den Admin, während er den Server konfiguriert, vor Angriffen zu schützen. Der Befehl kann natürlich auch verwendet werden, um unerlaubt ins Spielgeschehen einzugreifen. Auch wenn es auf den ersten Blick lustig erscheint, sollte man von der Nutzung absehen. Die anderen Spieler lassen sich schnell verärgern und verlassen den Server.

Der Spielername muss ausgeschrieben werden.

Beispiele:

```
admin_godmode "I am GOD!" "on"
admin_godmode "Kanonenfutter" "off"
```

Im ersten Fall wird der Spieler „I am GOD“ in die Lage versetzt, unsterblich zu sein, während im zweiten Fall dem Spieler „Kanonenfutter“ diese Fähigkeit aberkannt wird.

Access-Level: 8192
Gehört zu: plugin_cheat

5.5.2. admin_noclip

`admin_noclip <Spielername> <"on" oder "off">`

Der Befehl admin_noclip erlaubt dem Admin, einem Spieler die Fähigkeit zu geben, durch Wände zu gehen. Eine serverweite Meldung wird automatisch abgesetzt, sobald der Befehl ausgeführt wird! Diese wird unabhängig von der Einstellung admin_quiet abgesetzt.

Eine sinnvolle Verwendung könnte etwa bei SvenCoop denkbar sein, wenn man beispielsweise an einer bestimmten Stelle wegen eines Mapfehlers hängen bleibt. Der Befehl kann natürlich auch verwendet werden, um unerlaubt ins Spielgeschehen einzugreifen. Allerdings sollte man genau wissen, wo man hinläuft...

Der Spielername muss ausgeschrieben werden.

Beispiele:
```
admin_noclip "I am a ghost!" "on"
admin_noclip "Beule" "off"
```

Im ersten Fall wird der Spieler „I am a ghost“ in die Lage versetzt, durch Wände zu gehen, während im zweiten Fall dem Spieler „Beule“ diese Fähigkeit aberkannt wird.

Access-Level: 8192
Gehört zu: plugin_cheat

5.5.3. admin_stack

`admin_stack`

Dieser Befehl stapelt alle Spieler oberhalb des Admins. Er kann insbesondere bei Clan-Trainings verwendet werden, um die undisziplinierten, marodierenden Clanmitglieder zusammenzubringen.

Es sollte dringend darauf geachtet werden, dass genügend Platz über einem ist. Ansonsten könnten einige Spieler in der Decke hängen bleiben.

Beispiel:
```
admin_stack
```

Stapelt alle Spieler über einem.

Access-Level: 8192
Gehört zu: plugin_cheat

5.5.4. admin_teleport

`admin_teleport <Spieler> <x> <y> <z>`

Mit diesem Befehl ist es möglich einen Spieler an eine bestimmte Stelle auf der Map zu teleportieren. Dafür benötigt man allerdings die genauen Koordinaten. Da diese über den Befehl admin_userorigin oder anderweitiges Wissen über die Map erworben werden müssen, ist der Befehl eher als Proof-of-Concept für die Teleport-Funktion anzusehen.

Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Beispiel:
`admin_teleport "Captain Jim Kirk" 100 1431 -10`

Der Spieler „Captain Jim Kirk“ wird an die Koordinaten $x = 100$, $y = 1431$ und $z = -10$ teleportiert. Wo immer das auch ist...

Access-Level: 8192
Gehört zu: plugin_cheat

Siehe auch:
admin_userorigin

5.5.5. admin_userorigin

`admin_userorigin <Spielername>`

Dieser Befehl zeigt die Koordinaten eines Spielers auf der Map an. Eigentlich ist er nur in Verbindung mit dem Teleportbefehl admin_teleport von Nutzen. Er ist also wie dieser nur als Proof-of-Concept anzusehen.

Der Spielername muss exakt angegeben werden.

Beispiel:
`admin_userorigin "Hide and Seek"`

Es werden die aktuellen Koordinaten des Spielers „Hide and Seek“ angezeigt.

Access-Level: 8192
Gehört zu: plugin_cheat

Siehe auch:
admin_teleport

5.6. plugin_CS

Das CS Plugin ist, wie der Name schon sagt, primär für die Administration der Counter-Strike Modifikation gedacht. Nur wenn man auch wirklich Counter-Strike auf dem Server laufen lässt, sollte man das Plugin laden. Alle anderen Modifikationen werden damit wenig anfangen können.

Es lassen sich mit dem Plugin einige CS-spezifische Servervariablen setzen. Es ist aber insbesondere auch die Möglichkeit von Waffenrestriktionen interessant. Um beispielsweise den Kauf des unbeliebten Schilds zu unterbinden, ist die Einbindung von plugin_CS Pflicht.

5.6.1. admin_autokick

`admin_autokick [0/1]`

Der Befehl „admin_autokick" erlaubt das setzen der Servervariable „mp_autokick". Aktiviert man diese Funktion, werden Teamkiller oder sogenannte Idler automatisch vom Server geworfen. Ein Teamkill kann auch versehentlich geschehen. Der Spieler wird dann aber trotzdem vom Server geworfen. Man sollte besser auf TK-Plugins zurückgreifen.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_autokick 1
admin_autokick 0
admin_autokick
```

Im ersten Beispiel wird der Autokick aktiviert, im zweiten deaktiviert. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

Siehe auch:
admin_tkpunish, admin_hpenalty

5.6.2. admin_autoteambalance

`admin_autoteambalance [0/1]`

Der Befehl „admin_autoteambalance“ verstellt im laufenden Betrieb die Servervariable „mp_autoteambalance“. Aktiviert man diese Funktion, wird beim Rundenwechsel eine etwaige Überzahl durch Verschieben von Spielern ins kleinere Team ausgelichen.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_autoteambalance 1
admin_autoteambalance 0
admin_autoteambalance
```

Im ersten Beispiel wird das Auto-Teambalance aktiviert, im zweiten deaktiviert. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

Siehe auch:
admin_limitteams

5.6.3. admin_buytimer

`admin_buytimer [Minuten]`

Der Befehl „admin_buytimer“ verstellt die Servervariable „mp_buytimer“. Hiermit kann man einstellen, wie viele Minuten nach dem Rundenstart noch eingekauft werden darf. Übliche Werte sind 0,25 Minuten (15 Sekunden) oder 0,5 Minuten (30 Sekunden).

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_buytimer 0.25
admin_buytimer 1
admin_buytimer
```

Im ersten Beispiel wird die Kaufzeit auf 15 Sekunden im zweiten auf 60 Sekunden gesetzt. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.4. admin_c4timer

`admin_c4timer [Sekunden]`

Der Befehl „admin_c4timer“ setzt die Servervariable „mp_c4timer“. Hiermit kann man einstellen, wie lang der Countdown des Zeitzünders der Bombe ist. Übliche Werte sind 45 Sekunden oder 35 Sekunden.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_c4timer 35
admin_c4timer 45
admin_c4timer
```

Im ersten Beispiel wird der Bombencountdown auf 35 Sekunden im zweiten auf 45 Sekunden gesetzt. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.5. admin_chattime

`admin_chattime [Sekunden]`

Dieser Befehl setzt die Servervariable „mp_c4timer“. Man kann damit festlegen, wie lange nach dem Ende der Map noch gechattet werden kann, bevor die neue Map geladen wird. Übliche Werte sind 5 Sekunden oder 10 Sekunden.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_chattime 5
admin_chattime 10
admin_chattime
```

Im ersten Beispiel wird die Chatzeit am Ende der Map auf 5 Sekunden im zweiten auf 10 Sekunden gesetzt. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.6. admin_consistency

`admin_consistency [0/1]`

Dieser Befehl setzt die Servervariable „mp_consistency“. Man legt damit fest, ob der Server spielerspezifische Texturen und Models ablehnen soll. Mit „1“ wird die Überprüfung aktiviert, mit „0“ deaktiviert.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_consistency 1
admin_consistency 0
admin_consistency
```

Im ersten Beispiel wird die Ablehnung von Custommaterial aktiviert im zweiten deaktiviert. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.7. admin_ct

`admin_ct <Spieler>`

Man kann mit diesem Befehl einen Spieler ins Counter-Terroristen (CT) Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben. Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden. Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiele:

```
admin_ct "Ich bin kein Terrorist!"
admin_ct Big
admin_ct STEAM_0:123456
admin_ct 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Ich bin kein Terrorist!“ zu den Counter-Terroristen verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) zu den Counter-Terroristen verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 128
Gehört zu: plugin_CS

Siehe auch:
allow_client_exec, admin_t

5.6.8. admin_fadetoblack

`admin_fadetoblack [0/1]`

Dieser Befehl setzt die Servervariable „mp_fadetoblack“. Stirbt man, wird der Bildschirm schwarz, so dass man das Spielgeschehen nicht mehr verfolgen kann. In einigen wenigen Ligen ist die Aktivierung (1) Pflicht.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_fadetoblack 1
admin_fadetoblack 0
admin_fadetoblack
```

Im ersten Beispiel wird das Schwarzwerden des Bildschirms beim Ableben aktiviert im zweiten deaktiviert. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.9. admin_flashlight

`admin_flashlight [0/1]`

Dieser Befehl verändert die Servervariable „mp_flashlight“. Sie erlaubt oder verbietet die Benutzung der Taschenlampe durch die Spieler. Aktiviert wird die Funktion durch „1“, deaktiviert wird sie durch „0“.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_flashlight 1
admin_flashlight 0
admin_flashlight
```

Im ersten Beispiel wird die Verwendung der Taschenlampe erlaubt im zweiten verboten. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.10. admin_footsteps

`admin_footsteps [0/1]`

Dieser Befehl verändert die Servervariable „mp_footsteps“. Sie unterbindet oder aktiviert die Hörbarkeit der Spielerschritte. Aktiviert wird die Funktion durch „1“, deaktiviert wird sie durch „0“.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_footsteps 1
admin_footsteps 0
admin_footsteps
```

Im ersten Beispiel werden die Schritte der Spieler hörbar im zweiten wird die Hörbarkeit unterbunden. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.11. admin_forcecamera

`admin_forcecamera [0/1/2]`

Dieser Befehl setzt die Servervariable „mp_forcecamera“. Man kann damit einstellen, welche Einblicke der Spieler hat, wenn er tot ist. Bei „0“ kann er alle Einblicke verwenden. Bei „1“ ist er auf sein eigenes Team und bei „2“ nur auch das eigene Team in der Chasecam beschränkt.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_forcecamera 0
admin_forcecamera 1
admin_forcecamera 2
admin_forcecamera
```

In den ersten drei Beispielen werden die Kameraeinblickmöglichkeiten beschränkt. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

Siehe auch:
admin_forcechasecam

5.6.12. admin_forcechasecam

`admin_forcechasecam [0/1/2]`

Dieser Befehl setzt die Servervariable „mp_forcechasecam“. Man kann damit einstellen, welche Blickwinkel der Spieler nach seinem Ableben hat. Bei „0“ kann man sich frei bewegen, bei „1“ ist man auf Spieler beschränkt und bei „2“ kann man dem Spieler nur in der Egoperspektive folgen.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_forcechasecam 0
admin_forcechasecam 1
admin_forcechasecam 2
admin_forcechasecam
```

In den ersten drei Beispielen werden die Kamerablickwinkel beschränkt. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

Siehe auch:
admin_forcecamera

5.6.13. admin_freezetime

`admin_freezetime [Sekunden]`

Dieser Befehl verändert die Servervariable „mp_freezetime“. Hiermit kann eingestellt werden, wie lange sich Spieler nach Rundenbeginn nicht bewegen können. Üblicherweise sind das 5 oder 6 Sekunden. Diese Zeit ist in der Regel ausreichend, um einzukaufen.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_freezetime 5
admin_freezetime 0
admin_freezetime
```

Im ersten Beispiel wird die Bewegungslosigkeit am Anfang der Runde auf 5 Sekunden gestellt. Im zweiten ist die Funktion abgeschaltet. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.14. admin_ghostfrequency

`admin_ghostfrequency [Sekunden]`

Dieser Befehl setzt die Servervariable „mp_ghostfrequency“. Es wird damit eingestellt, wie häufig „Geister“ (z.B. Spectator oder Tote) eine Aktualisierung bekommen. Die Standardeinstellung ist „0.1“.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_ghostfrequency 0.1
admin_ghostfrequency 1
admin_ghostfrequency
```

Im ersten Beispiel wird die Aktualsierung auf 100 ms gestellt. Im zweiten Fall bekommen die „Geister“ nur jede Sekunde ein Update. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.15. admin_hpenalty

`admin_hpenalty [Anzahl]`

Dieser Befehl verändert die Servervariable „mp_hostagepenalty“. Man stellt damit ein, wieviele Geiseln ein Spieler töten darf, bevor er vom Server geworfen wird.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
```
admin_hpenalty 2
admin_hpenalty 0
admin_hpenalty
```

Im ersten Beispiel ist dem Spieler erlaubt, zwei Geiseln zu töten, während im zweiten die Überwachung abgeschaltet ist. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.16. admin_kickpercent

`admin_kickpercent [Anteil]`

Dieser Befehl setzt die Servervariable „mp_kickpercent“. Man stellt damit ein, wieviele Spieler prozentual für den Kick eines anderen Spielers voten müssen. Dies geschieht über `vote <Session ID>` in der Console. Leider ist die Anwendung dieser Funktion für die meisten Spieler ein Buch mit sieben Siegeln. Es handelt sich bei „vote“ um keine Admin Mod Funktionalität.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_kickpercent 1
admin_kickpercent 0.4
admin_kickpercent
```

Im ersten Beispiel müssten alle Spieler einem Kick zustimmen, während im zweiten nur 40% benötigt würden. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.17. admin_limitteams

`admin_limitteams [Anzahl]`

Dieser Befehl verändert die Servervariable „mp_limitteams“. Hiermit wird definiert wie groß der Überhang in der Spielerzahl zwischen den Teams maximal sein darf. „0“ schaltet die Funktion ab.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_limitteams 2
admin_limitteams 0
admin_limitteams
```

Im ersten Beispiel darf ein Team maximal zwei Spieler mehr als das andere haben, während im zweiten keine Einschränkung vorhanden ist. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

Siehe auch:
admin_autoteambalance

5.6.18. admin_maxrounds

`admin_maxrounds [Rundenzahl]`

Dieser Befehl ermöglicht es, die Servervariable „mp_maxrounds“ zu verändern. Man stellt ein, nach wievielen Runden spätestens ein Rundenwechsel stattfindet. Abschalten lässt sich die Funktion, in dem man „0“ eingibt.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_maxrounds 30
admin_maxrounds 0
admin_maxrounds
```

Im ersten Beispiel wird spätestens nach 30 Runden die Map gewechselt, während im zweiten die Map nicht auf Basis der Rundenzahl gewechselt wird. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.19. admin_mapvoteratio

`admin_mapvoteratio [Anteil]`

Dieser Befehl setzt die Servervariable „mp_mapvoteratio“. Man stellt damit ein, wieviele Spieler prozentual für eine Map voten müssen, damit zu dieser gewechselt wird. Dies geschieht über `votemap <Mapname>"` in der Console. Leider ist die Anwendung dieser Funktion für die meisten Spieler ein Buch mit sieben Siegeln. Es handelt sich bei „votemap“ um keine Admin Mod Funktionalität.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_mapvoteratio 1
admin_mapvoteratio 0.4
admin_mapvoteratio
```

Im ersten Beispiel müssten alle Spieler für eine bestimmte Map votieren, während im zweiten nur 40% benötigt würden. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.20. admin_playerid

`admin_playerid [Rundenzahl]`

Dieser Befehl ermöglicht es, die Servervariable „mp_playerid“ zu verändern. Es wird eingestellt, ob man den Namen jedes Spielers im Fadenkreuz zu sehen bekommt (0), nur den der Teammitglieder (1) oder von niemanden (2).

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
`admin_playerid 0`
`admin_playerid 1`
`admin_playerid`

Im ersten Beispiel kann man den Namen der Teammitglieder und den der Gegner beim Zielen lesen, während dies im zweiten Fall auf die Teammitglieder beschränkt wird. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.21. admin_restart

`admin_restart <Sekunden>`

Dieser Befehl ermöglicht es, die Servervariable „mp_restartround“ zu verändern. Man aktiviert damit einen Countdown, der einen Mapreset durchführt (z.B. bei Clanwars).

Ein synonymer Befehl ist admin_restartround.

Beispiel:
`admin_restart 5`

Nach Eingabe startet die Map nach 5 Sekunden neu.

Access-Level: 2
Gehört zu: plugin_CS

Siehe auch:
admin_restartround, admin_vote_restart

5.6.22. admin_restartround

`admin_restartround <Sekunden>`

Dieser Befehl ermöglicht es, die Servervariable „mp_restartround“ zu verändern. Man aktiviert damit einen Countdown, der einen Mapreset durchführt (z.B. bei Clanwars).

Ein synonymer Befehl ist admin_restart.

Beispiel:
`admin_restartround 5`

Nach Eingabe startet die Map nach 5 Sekunden neu.

Access-Level: 2
Gehört zu: plugin_CS

Siehe auch:
admin_restart, admin_vote_restart

5.6.23. admin_restrict

`admin_restrict [Option]`

Der Befehl „admin_restrict“ ermöglicht es, Waffen und/oder Equipment auf dem Server zu verbieten. Dabei hilft eine Vielzahl an Optionen, die Einschränkungen bis ins kleinste Detail einzustellen.

Aufgesammelte Waffen und Equipment, die sich bereits im Besitz der Spieler befinden, werden nicht erkannt und verschwinden erst aus dem Spiel, wenn sie nach Rundenende nicht mehr am Mann sind.

Gibt man keine Option an, werden die aktuellen Restriktionen angezeigt. Diese Funktion steht allen Spielern zur Verfügung (Recht 0).

Für temporäre Einstellungen braucht man das Recht 32, zum Speichern und Löschen von Einstellungen wird jedoch das Recht 512 benötigt.

Die Einstellungen können über den Befehl admin_unrestrict rückgängig gemacht werden. Darüber hinaus steht ein Menü über den Befehl admin_restrictmenu zur Verfügung.

Optionen:
`<on/off>`
Schaltet die Restriktionen ein oder aus.

`save [default/map]`
Speichert die aktuellen Restriktionen allgemein oder nur für die aktuelle Map.

`delete [default/map]`
Löscht die aktuellen, allgemeinen oder mapspezifischen Restriktionen.

restore
Stellt die allgemeinen und mapspezifischen Restriktionen wieder her.

Die folgenden Optionen ermöglichen neben dem serverweiten Verbot auch optional die Beschränkung eines Teams oder eines bestimmten Spielers.

[player <Name>/team <ct/t>] all
Verbietet jeglichen Einkauf.

[player <Name>/team <ct/t>] weapons
Verbietet den Kauf aller Waffen.

[player <Name>/team <ct/t>] menu [Menüname/-nummer]
Verbietet ein bestimmtes Kaufmenü über die Menünummer.

[player <Name>/team <ct/t>] [Name Waffe/Equipment]
Verbietet eine Waffe oder einen Gegenstand auf Basis seines Namens.

[player <Name>/team <ct/t>] <Menünummer> <Nummer Waffe/Equipment>
Verbietet eine Waffe oder einen Gegenstand auf Basis der Menü- und Auswahlnummer.

help
Gibt Hinweise zur Anwendung des Befehls „admin_restrict“.

Beispiele:
admin_restrict
Es werden die aktuellen Restriktionen angezeigt.

admin_restrict on
Die Restriktionen werden eingeschaltet.

admin_restrict save default
Die aktuell gültigen Restriktionen werden für alle Maps gespeichert, die keine Sonderregeln haben.

admin_restrict save map
Die aktuell gültigen Restriktionen werden nur für die aktuelle Map gespeichert (Sonderregel).

admin_restrict delete map
Die Restriktionen für die aktuelle Map werden gelöscht (Sonderregel). Es gelten dann die allgemeinen Restriktionen, sofern diese definiert sind.

admin_restrict restore
Die gespeicherten Restriktionen zur gespielten Map werden wiederhergestellt. Alle seit Mapstart getätigten und nicht gespeicherten Restriktionen werden rückgängig gemacht. Dies ist in der Regel einfacher als den Befehl admin_unrestrict zu bemühen.

admin_restrict all
Verbietet allen Spielern beider Teams den Einkauf.

`admin_restrict player ‘Back to the roots’ weapons`
Verbietet dem Spieler „Back to the roots“ den Kauf einer Waffe. Die einfachen Anführungszeichen sind bei Spielernamen mit Leerzeichen zu verwenden. Alternativ wird auch ein eindeutiger Teil des Namens akzeptiert.

`admin_restrict team ct menu 1`
`admin_restrict team ct menu pistols`
Beide Schreibweisen verbieten dem CT-Team den Kauf von Pistolen.

`admin_restrict player ‘Will kaufen’`
`admin_restrict team t`
Das erste Beispiel zeigt die aktuellen Restriktionen für den Spieler „Will kaufen“, das zweite hingegen die aktuellen Restriktionen des Terror-Teams.

`admin_restrict Magnum Sniper Rifle`
`admin_restrict awp`
`admin_restrict 4 6`
Alle drei Beispiele verbieten die AWP. Letzteres über die Menünummer und die Auswahlnummer.

Access-Level: 0, 32, 512
Gehört zu: plugin_CS

Siehe auch:
admin_restrictmenu, admin_unrestrict

5.6.24. admin_restrictmenu

`admin_restrictmenu`

Der Befehl „admin_restrictmenu“ öffnet ein Menü, mit dem man bequem Waffen und/ oder Equipment auf dem Server verbieten kann. Es ist dadurch prinzipbedingt einfacher zu bedienen als admin_restrict oder admin_unrestrict.

Für die Verwendung muss `amv_enable_beta "menu1"` in der adminmod.cfg gesetzt sein.

Aufgesammelte Waffen und Equipment, die sich bereits im Besitz der Spieler befinden, werden nicht erkannt und verschwinden erst aus dem Spiel, wenn sie nach Rundenende nicht mehr am Mann sind.

Gibt man keine Option an, werden die aktuellen Restriktionen angezeigt. Diese Funktion steht allen Spielern zur Verfügung (Recht 0).

Für temporäre Einstellungen braucht man das Recht 32, zum Speichern und Löschen von Einstellungen wird jedoch das Recht 512 benötigt.

Beispiel:
`admin_restrictmenu`

Es wird ein Menü zum Verbieten von Waffen und/oder Equipment aufgerufen.

Access-Level: 0, 32, 512
Gehört zu: plugin_CS

Siehe auch:
admin_restrict, admin_unrestrict, admin_enable_beta

5.6.25. admin_roundtime

`admin_roundtime [Minuten]`

Dieser Befehl ermöglicht es, die Servervariable „mp_roundtime“ zu verändern. Es wird eingestellt, wie lange eine Runde maximal dauert (z.B. 3 Minuten).

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_roundtime 3
admin_roundtime 1.75
admin_roundtime
```

Im ersten Beispiel wird die Rundenzeit auf 3 Minuten gesetzt, während im zweiten Fall die Rundenzeit 1:45 Minuten beträgt. Im letzten wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.26. admin_startmoney

`admin_startmoney [Dollar]`

Dieser Befehl setzt die Servervariable „mp_startmoney“. Üblicherweise beginnt jeder Spieler nach dem Mapstart oder beim Betreten des Servers mit $800. Hiermit ist es möglich, das Startgeld im Bereich von $0 bis $16000 zu verändern.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_startmoney 8000
admin_startmoney 800
admin_startmoney
```

Im ersten Beispiel wird das Startgeld mit $8000 festgelegt, während im zweiten Fall das Startgeld auf die üblichen $800 gesetzt wird. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.27. admin_t

`admin_t <Spieler>`

Man kann mit diesem Befehl einen Spieler ins Terroristen (T) Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben. Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden. Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiel:

```
admin_t "Ich bin kein CT!"
admin_t Big
admin_t STEAM_0:123456
admin_t 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Ich bin kein CT!“ zu den Terroristen verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) zu den Terroristen verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 128
Gehört zu: plugin_CS

Siehe auch:
allow_client_exec, admin_ct

5.6.28. admin_tkpunish

`admin_tkpunish [0/1]`

Mit „admin_tkpunish“ kann man die Servervariable „mp_tkpunish“ editieren. Man stellt damit ein, ob ein Teamkiller in der nächsten Runde aussetzen muss oder nicht. Das hält aber nicht wirklich irgendjemanden davon ab. Besser ist es ein TK-Plugin zu installieren.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:
`admin_tkpunish 1`
`admin_tkpunish 0`
`admin_tkpunish`

Im ersten Beispiel wird das Aussetzen bei einem Teamkill aktiviert, während im zweiten Beispiel die Funktion deaktiviert wird. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.6.29. admin_unrestrict

`admin_unrestrict [Option]`

Der Befehl „admin_unrestrict“ ermöglicht es, Waffen und/oder Equipment auf dem Server wieder zu erlauben. Dabei hilft eine Vielzahl an Optionen, diese Einschränkungen bis ins kleinste Detail einzustellen.

Die Einstellungen können über den Befehl admin_restrict rückgängig gemacht werden. Darüber hinaus steht ein Menü über den Befehl admin_restrictmenu zur Verfügung.

Die folgenden Optionen ermöglichen neben der serverweiten Erlaubnis auch optional den Zugriff eines Teams oder eines bestimmten Spielers auf die Gegenstände.

Optionen:
`[player <Name>/team <ct/t>] all`
Erlaubt den Einkauf aller Waffen und Gegenstände.

`[player <Name>/team <ct/t>] weapons`
Erlaubt den Kauf aller Waffen.

`[player <Name>/team <ct/t>] menu [Menüname/-nummer]`
Erlaubt ein bestimmtes Kaufmenü über die Menünummer.

`[player <Name>/team <ct/t>] [Name Waffe/Equipment]`
Erlaubt eine Waffe oder einen Gegenstand auf Basis seines Namens.

`[player <Name>/team <ct/t>] <Menünummer> <Nummer Waffe/Equipment>`
Erlaubt eine Waffe oder einen Gegenstand auf Basis der Menü- und Auswahlnummer.

Beispiele:
`admin_unrestrict all`
Erlaubt allen Spielern beider Teams den Einkauf.

`admin_unrestrict player 'Full metal jacket' weapons`
Erlaubt dem Spieler „Full metal jacket" den Kauf einer Waffe. Die einfachen Anführungszeichen sind bei Spielernamen mit Leerzeichen zu verwenden. Alternativ wird auch ein eindeutiger Teil des Namens akzeptiert.

`admin_unrestrict team ct menu 1`
`admin_unrestrict team ct menu pistols`
Beide Schreibweisen erlauben dem CT-Team den Kauf von Pistolen.

`admin_unrestrict Magnum Sniper Rifle`
`admin_unrestrict awp`
`admin_unrestrict 4 6`
Alle drei Beispiele erlauben die AWP. Letzteres über die Menünummer und die Auswahlnummer.

Access-Level: 32
Gehört zu: plugin_CS

Siehe auch:
admin_restrict, admin_restrictmenu

5.6.30. admin_vote_restart

`admin_vote_restart <Sekunden>`

Dieser Befehl löst einen Vote aus, ob die Servervariable „mp_restartround" verändert werden soll. Entscheidet sich die Mehrheit (basierend auf „map_ratio") dafür, wird die Map nach der festgelegten Zeit neu gestartet.

Beispiel:
`admin_vote_restart 5`

Entscheidet sich im Anschluss die Mehrheit der Spieler auf dem Server dafür, wird die Map 5 Sekunden nach Abschluss des Votes neu gestartet.

Access-Level: 1
Gehört zu: plugin_CS

Siehe auch:
admin_restart, admin_restartround, map_ratio

5.6.31. admin_winlimit

`admin_winlimit [Anzahl]`

Dieser Befehl verändert die Servervariable „mp_winlimit“. Wenn eines der Teams die gesetzte Anzahl an Runden gewinnt, wird die Map gewechselt.

Gibt man keine Zahl ein, wird die aktuelle Einstellung angezeigt. Diese Funktion steht allen Spielern offen.

Beispiele:

```
admin_winlimit 30
admin_winlimit 0
admin_winlimit
```

Im ersten Beispiel wird die Anzahl der gewonnenen Runden bis zum Mapwechsel auf 30 gesetzt, während im zweiten die Funktion abgeschaltet wird. Im letzten Beispiel wird die aktuelle Einstellung gezeigt.

Access-Level: 0, 512
Gehört zu: plugin_CS

5.7. plugin_fun

Das Fun-Plugin bringt eigentlich nur einige recht sinnfreie Funktionen mit sich. Grundsätzlich muss man dieses Plugin nicht mitladen lassen. Die Glow-Funktion können alle Spieler benutzen, sobald der Fun-Mode aktiviert wurde. Dies ist meist nicht im Interesse des Admins. Außerdem beinhaltet das Plugin ein Easter Egg, das ggf. auch nerven kann. Interessant hingegen könnte höchstens admin_fun sein, um die Glow- und Discofunktion temporär abzuschalten.

Im Zweifel einfach das Plugin auskommentieren.

5.7.1. admin_disco

`admin_disco`

Der Befehl schaltet den Discomodus ein bzw. auch wieder aus. Discomodus bedeutet, dass alle Spieler in unterschiedlichen Farben glühen und hin und wieder einige Centersays eingeblendet werden. Dafür muss der Fun-Mode aktiviert sein (admin_fun_mode). Das probiert man einmal aus, und dann bleibt die Funktion zumeist ungenutzt.

Beispiel:
`admin_disco`

Läuft der Discomodus nicht, wird er damit aktiviert. Ist der Discomodus jedoch aktiviert gewesen, ist er nun abgeschaltet.

Access-Level: 8192
Gehört zu: plugin_fun

Siehe auch:
admin_fun_mode

5.7.2. admin_fun

`admin_fun [on]`

Der Befehl schaltet die Glow- und Discofunktion an oder aus. Es wird dabei die Variable admin_fun_mode gesetzt und ggf. laufende Glows abgeschaltet. Die Einstellungen bleiben bis zum nächsten Aufruf der adminmod.cfg aktiv. Je nach Einstellung (mapchangecfgfile) kann das beim Mapwechsel oder beim nächsten Serverrestart sein. Alles, was nicht „on“ entspricht wird als „off“ interpretiert.

Beispiele:

```
admin_fun on
admin_fun off
admin_fun
```

Das erste Beispiel schaltet den Fun-Mode an. Die anderen Beispiele schalten den Fun-Mode und damit ggf. laufende Glows bis auf Weiteres ab.

Access-Level: 8192
Gehört zu: plugin_fun

Siehe auch:
admin_fun_mode

5.7.3. admin_glow

`admin_glow <Farbe/off>`

Mit diesem Befehl kann man sich selbst in einer festgelegten Farben glühen lassen oder den Effekt abschalten. Dafür muss der Fun-Mode aktiviert sein (admin_fun_mode).

Erlaubte Farben: red, blue, green, white, yellow, purple

Beispiele:

```
admin_glow green
admin_glow red
admin_glow off
```

Das erste Beispiel lässt einen in grün und das zweite in rot glühen. Das letzte Beispiel schaltet das Glühen bei einem selbst ab.

Access-Level: 8192
Gehört zu: plugin_fun

Siehe auch:
admin_fun_mode, say glow

5.7.4. say glow

`say glow <Farbe/off>`

Mit diesem Befehl kann man sich selbst in einer festgelegten Farben glühen lassen oder den Effekt abschalten. Dafür muss der Fun-Mode aktiviert sein (admin_fun_mode).

Erlaubte Farben: red, blue, green, white, yellow, purple

Beispiele:

```
say glow green
say glow red
say glow off
```

Das erste Beispiel lässt einen in grün und das zweite in rot glühen. Das letzte Beispiel schaltet das Glühen bei einem selbst ab.

Access-Level: 0
Gehört zu: plugin_fun

Siehe auch:
admin_fun_mode, admin_glow

5.8. plugin_hldsld_mapvote

Selbst in Zeiten vor Admin Mod gab es noch ein paar andere Administrationsprogramme für den HLDS. Dazu zählte auch hlds_ld (oder später halfd[2]). Es basierte auf der Skriptsprache TCL[3] und reagierte auf Logeinträge. Dieses bot ein für damalige Verhältnisse ausgefeiltes Mapvoting-System, das mit dem plugin_hldsld_mapvote in Admin Mod nachempfunden wurde.

Inzwischen gibt es wesentlich ausgefeiltere Mapvoting-Plugins für Admin Mod, die auch menübasierte Auswahlmöglichkeiten anbieten. Das Plugin wäre bei nächster Gelegenheit ausgetauscht worden, ein Prototyp existierte bereits. Eine neue Admin Mod Version hat es jedoch nicht mehr gegeben.

5.8.1. admin_cancelvote

`admin_cancelvote`

Dieser Befehl bricht einen eventuell laufenden Mapvote ab. Die gleiche Funktion lässt sich auch über den Chat (say cancelvote) ausführen.

Beispiel:
`admin_cancelvote`

In diesem Beispiel wird der laufende Mapvote abgebrochen. Falls keiner läuft, wird eine Fehlermeldung ausgegeben.

Access-Level: 2
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
say cancelvote

[2] `http://sourceforge.net/projects/halfd/`
[3] `http://www.tcl.tk/`

5.8.2. admin_denymap

`admin_denymap <Map>`

Dieser Befehl löscht alle Votes für die angegebene Map. Die gleiche Funktion lässt sich auch über den Chat (say denymap) ausführen.

Beispiel:
`admin_denymap de_dust2`

In diesem Beispiel werden alle Votes für die so häufig gespielte Map de_dust2 entfernt, um zu verhindern, dass zu dieser Map gewechselt wird.

Access-Level: 2
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
say denymap

5.8.3. admin_startvote

`admin_startvote`

Dieser Befehl startet einen Mapvote. Die gleiche Funktion lässt sich auch über den Chat (say mapvote oder say rockthevote) ausführen. Einige Einstellungen z.B. zur Dauer des Votes, der Häufigkeit etc. sind in der adminmod.cfg einstellbar.

Beispiel:
`admin_startvote`

In diesem Beispiel wird ein Mapvote gestartet.

Access-Level: 1
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
say mapvote, say rockthevote, admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration

5.8.4. say cancelvote

`say cancelvote`

Dieser Befehl bricht einen eventuell laufenden Mapvote ab. Die gleiche Funktion lässt sich auch über die Console (admin_cancelvote) ausführen.

Beispiel:
`say cancelvote`

In diesem Beispiel wird der laufende Mapvote abgebrochen. Falls keiner läuft, wird eine Fehlermeldung ausgegeben.

Access-Level: 2
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
admin_cancelvote

5.8.5. say denymap

`say denymap <Map>`

Dieser Befehl löscht alle Votes für die angegebene Map. Die gleiche Funktion lässt sich auch über die Console (admin_denymap) ausführen.

Beispiel:
`say denymap de_dust2`

In diesem Beispiel werden alle Votes für die so häufig gespielte Map de_dust2 entfernt, um zu verhindern, dass zu dieser Map gewechselt wird.

Access-Level: 2
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
admin_denymap

5.8.6. say mapvote

`say mapvote`

Dieser Befehl startet einen Mapvote. Die gleiche Funktion lässt sich auch über say rockthevote oder die Console (admin_startvote) ausführen. Einige Einstellungen z.B. zur Dauer des Votes, der Häufigkeit etc. sind in der adminmod.cfg einstellbar.

Beispiel:
`say mapvote`

In diesem Beispiel wird ein Mapvote gestartet.

Access-Level: 1
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
say rockthevote, admin_startvote, admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration

5.8.7. say rockthevote

`say rockthevote`

Dieser Befehl startet einen Mapvote. Die gleiche Funktion lässt sich auch über say mapvote oder die Console (admin_startvote) ausführen. Einige Einstellungen z.B. zur Dauer des Votes, der Häufigkeit etc. sind in der adminmod.cfg einstellbar.

Beispiel:
`say rockthevote`

In diesem Beispiel wird ein Mapvote gestartet.

Access-Level: 1
Gehört zu: plugin_hldsld_mapvote

Siehe auch:
say mapvote, admin_startvote, admin_vote_autostart, admin_vote_echo, admin_vote_freq, admin_vote_maxextend, admin_vote_ratio, amv_vote_duration

5.8.8. say vote

```
say vote <Map>
```

Mit diesem Befehl kann man sich während eines laufenden Mapvotes für eine Map entscheiden. Sofern sich genügend Personen für eine Map finden, wird zu dieser gewechselt.

Beispiel:

```
say vote cs_office
```

In diesem Beispiel wird ein Vote für die Map cs_office abgegeben. Finden sich genügend Mitstreiter dafür, wird zu ihr gewechselt. Bekommt keine Map die Mehrheit wird der Mapcycle aktiv.

Access-Level: 1
Gehört zu: plugin_hldsld_mapvote

5.9. plugin_message

Das kleinste der Standardplugins beinhaltet keine Befehle. Hier wird lediglich die Willkommensmeldung (admin_connect_msg) und die sich wiederholende, mittig positionierte Meldung (admin_repeat_freq und admin_repeat_msg) angezeigt.

5.10. plugin_retribution

Ich habe bereits an anderer Stelle in diesem Kompendium angemerkt, dass der Admin durchaus Möglichkeiten besitzt, einen Störer auf dem Server angemessen zu bestrafen. Dazu muss man niemandem die Konfiguration zerstören.

Das Retribution-Plugin beinhaltet diverse Befehle, um einen Spieler zu schlagen, zu töten, einzugraben oder einfach mundtot zu machen. Auch kann man den Spieler Clientbefehle ausführen lassen.

5.10.1. admin_bury

`admin_bury <Spielername>`

Mit diesem Befehl kann man einen Spieler in den Boden versenken, so dass er sich nicht mehr rühren kann. Der Spielername muss exakt eingegeben werden.

Das funktioniert nur sicher, wenn der Spieler beim Absetzen des Befehls sich nicht im Sprung befindet. Es handelt sich um eine Teleport-Funktion, die den Spieler in den Boden versetzt. Darüber hinaus behält der Spieler seine Waffe, so dass er nicht wehrlos ist. Zudem ist zu beachten, dass unter bestimmten Umständen der Spieler dabei schlecht gesehen werden kann. Auch kann sich der Spieler durch einen Selbst-Kill aus der Lage befreien.

Vorausgesetzt admin_fx ist in der adminmod.cfg aktiviert, wird der Befehl mit einem Sound untermalt.

Diese Aktion kann durch den Befehl admin_unbury rückgängig gemacht werden. Beispiel:
`admin_bury Kriegsbeil`

In diesem Beispiel wird der Spieler „Kriegsbeil" vergraben...

Access-Level: 8192
Gehört zu: plugin_retribution

Siehe auch:
admin_unbury

5.10.2. admin_execall

`admin_execall <Befehl>`

Der Befehl admin_execall führt bei allen Spielern den gewünschten Befehl aus. Dafür muss auch allow_client_exec bzw. bei Bots admin_bot_protection aktiviert worden sein. Befehle werden zudem nicht an einen HLTV weitergeleitet.

Die Ausführung einiger Client-Befehle wird von Admin Mod selber unterbunden, um Spieler vor Amok laufenden Admins zu schützen.

Beispiel:
`admin_execall cd eject`

Dies fährt bei alle nicht geschützten Spielern die Lade des CD/DVD-Laufwerks aus.

Access-Level: 65536
Gehört zu: plugin_retribution

Siehe auch:
admin_execclient, admin_execteam, allow_client_exec

5.10.3. admin_execclient

`admin_execclient <ID/IP/Name> <Befehl>`

Der Befehl admin_execclient führt beim angegebenen Spieler den gewünschten Befehl aus. Es muss allow_client_exec aktiviert bzw. bei Bots admin_bot_protection deaktiviert worden sein. Befehle werden zudem nicht an einen HLTV weitergeleitet.

Die Ausführung einiger Client-Befehle wird von Admin Mod selber unterbunden, um Spieler vor Amok laufenden Admins zu schützen.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Beispiele:

```
admin_execlient "Need Coffee" cd eject
admin_execlient Big cd eject
admin_execlient STEAM_0:123456 cd eject
admin_execlient 23.156.43.12 cd eject
```

Im ersten Beispiel fährt die Lade des CD/DVD-Laufwerks des Spielers „Need Coffee“ aus, sofern dieser nicht immun dagegen ist. Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 65536
Gehört zu: plugin_retribution

Siehe auch:
admin_execall, admin_execteam, allow_client_exec

5.10.4. admin_execteam

`admin_execteam <Teamnummer> <Befehl>`

Der Befehl admin_execteam führt bei allen Spielern eines Teams den gewünschten Befehl aus. Um den Befehl überhaupt ausführen zu können, muss auch allow_client_exec aktiviert bzw. bei etwaigen Bots admin_bot_protection deaktiviert worden sein. Befehle werden zudem nicht an einen HLTV weitergeleitet. Voraussetzung ist natürlich, dass es bei der Modifikation überhaupt Teams gibt.

Die Ausführung einiger Client-Befehle wird von Admin Mod selber unterbunden, um Spieler vor Amok laufenden Admins zu schützen.

Beispiel:

```
admin_execteam 1 cd eject
```

Dies fährt bei allen nicht geschützten Spielern des Teams die Lade des CD/DVD-Laufwerks aus. Bei Counter-Strike würde dies das Terror-Team bei TFC das blaue Team betreffen.

Access-Level: 65536
Gehört zu: plugin_retribution

Siehe auch:
admin_execall, admin_execclient, allow_client_exec

5.10.5. admin_gag

`admin_gag <ID/IP/Name> [Minuten]`

Mit diesem Befehl kann man einen Spieler mundtot machen (knebeln). Er kann dann den Chat nicht mehr verwenden. Sämtliche Nachrichten werden geblockt. Der Voicechat ist davon nicht betroffen.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Diese Aktion kann durch den Befehl admin_ungag rückgängig gemacht werden.

Ob der Spieler mit dem eigenen Team sprechen kann (admin_gag_sayteam), oder ob er seinen Namen ändern darf (admin_gag_name), hängt von Einstellungen in der adminmod.cfg ab. Bei letzterem ist darüber hinaus auch allow_client_exec notwendig.

Beispiele:
```
admin_gag Kindergeburtstag 5
admin_gag Kindergeburtstag
admin_gag Big
admin_gag STEAM_0:123456
admin_gag 23.156.43.12
```

Im ersten Beispiel wird der Spieler „Kindergeburtstag“ im Chat für 5 Minuten im zweiten Beispiel für immer zum Schweigen gebracht. Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 2048
Gehört zu: plugin_retribution

Siehe auch:
admin_ungag, admin_gag_name, admin_gag_sayteam, allow_client_exec

5.10.6. admin_llama

`admin_llama <ID/IP/Name>`

Mit diesem Befehl kann man einen Spieler zum Lama erklären. Er wird auch in „Llama" umbenannt. Alle seine Chatnachrichten werden durch „Ooorgle", „Bleeeat!" oder „Brawwrr!" ersetzt. Der Voicechat ist davon nicht betroffen. Um den Befehl ausführen zu können, muss allow_client_exec aktiviert werden.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Diese Aktion kann durch den Befehl admin_unllama rückgängig gemacht werden.

Beispiel:

```
admin_llama Kindergeburtstag
admin_llama Big
admin_llama STEAM_0:123456
admin_llama 23.156.43.12
```

Im ersten Beispiel wird der Spieler „Kindergeburtstag" in „Llama" umbenannt, und gibt bei jeder seiner Chatnachrichten Unverständliches von sich. Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big" oder „Big" im Namen (sofern kein anderer Spieler „Big" im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_retribution

Siehe auch:
admin_unllama, allow_client_exec

5.10.7. admin_slap

`admin_slap <ID/IP/Name>`

Mit diesem Befehl kann man einem Spieler einen Schlag versetzen, der ihn 5 Lebenspunkte kostet, und ihn ein wenig aus der Bahn wirft.

Führt man den Befehl mehrmals kurz hintereinander aus, kann es passieren, dass der Spieler durch die halbe Map fliegt. Er verliert natürlich bei jedem Schlag weitere 5 Lebenspunkte. Mit admin_slap wird die Lebenspunktzahl jedoch nie niedriger als einen Lebenspunkt. Ein Spieler kann also nicht direkt durch ein Slap sterben. Das passiert höchstens, wenn er z.B. mit sehr wenigen Lebenspunkten durch den Schubser zu tief fällt.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Beispiel:
```
admin_slap SM
admin_slap Big
admin_slap STEAM_0:123456
admin_slap 23.156.43.12
```

Im ersten Beispiel steckt der Spieler „SM“ ein wenig Prügel ein. Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 128
Gehört zu: plugin_retribution

5.10.8. admin_slay

`admin_slay <ID/IP/Name>`

Mit diesem Befehl lässt man einen Spieler das Zeitliche segnen.

Vorausgesetzt admin_fx ist in der adminmod.cfg aktiviert, wird der Befehl mit einem Sound untermalt.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Man kann die Aktion auch auf ein ganzes Team anwenden, in dem man den Befehl admin_slayteam verwendet.

Beispiel:
```
admin_slay Killme
admin_slay Big
admin_slay STEAM_0:123456
admin_slay 23.156.43.12
```

Im ersten Beispiel segnet der Spieler „Killme“ das Zeitliche. Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 128
Gehört zu: plugin_retribution

Siehe auch:
admin_slayteam

5.10.9. admin_slayteam

`admin_slayteam <Teamnummer>`

Mit diesem Befehl lässt man alle Spieler des angegebenen Teams das Zeitliche segnen.

Vorausgesetzt admin_fx ist in der adminmod.cfg aktiviert, wird der Befehl mit einem Sound untermalt.

Man kann die Aktion auch auf einen einzelnen Spieler anwenden, in dem man den Befehl admin_slay verwendet.

Beispiel:
`admin_slayteam 2`

Dies lässt alle nicht geschützten Spieler des Teams das Zeitliche segnen. Bei Counter-Strike würde dies das CT-Team bei TFC das rote Team betreffen.

Access-Level: 128
Gehört zu: plugin_retribution

Siehe auch:
admin_slay

5.10.10. admin_unbury

`admin_unbury <Spielername>`

Mit diesem Befehl kann man einen ursprünglich im Boden versenkten Spieler wieder von seinem Leiden befreien.

Der Spielername muss exakt eingegeben werden.

Beispiel:

```
admin_unbury Klappstuhl
admin_unbury Big
admin_unbury STEAM_0:123456
admin_unbury 23.156.43.12
```

Im ersten Beispiel wird der Spieler „Klappstuhl“ wieder ausgegraben... Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_retribution

Siehe auch:
admin_bury

5.10.11. admin_ungag

`admin_ungag <ID/IP/Name>`

Mit diesem Befehl kann man einen ursprünglich geknebelten Spieler wieder von seinem Leiden befreien, und damit den Chat wieder zu benutzen. Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Beispiel:
```
admin_ungag Kindergeburtstag
admin_ungag Big
admin_ungag STEAM_0:123456
admin_ungag 23.156.43.12
```

Im ersten Beispiel wird dem Spieler „Kindergeburtstag“ wieder erlaubt, den Chat zu verwenden.

Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt.

Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 2048
Gehört zu: plugin_retribution

Siehe auch:
admin_gag

5.10.12. admin_unllama

`admin_unllama <ID/IP/Name>`

Mit diesem Befehl kann man einen ursprünglich in ein Lama verwandelten Spieler wieder von seinem Leiden befreien.

Es kann sowohl die Steam ID, eine IP oder ein Name angegeben werden. Beim Namen genügt auch ein eindeutiger Teil des Namens.

Beispiel:
```
admin_unllama Kindergeburtstag
admin_unllama Big
admin_unllama STEAM_0:123456
admin_unllama 23.156.43.12
```

Im ersten Beispiel wird der Spieler „Kindergeburtstag“ wieder erlaubt, im Chat Verständliches zu schreiben.

Im nächsten Beispiel wird dies beim Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ausgeführt.

Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_retribution

Siehe auch:
admin_llama

5.11. plugin_spawn

Es ist leider nicht bekannt, ob admin_listspawn, admin_movespawn, admin_removespawn und admin_spawn jemals funktioniert haben. In jedem Fall sind die Befehle funktionslos. Das Spawn-Plugin zu laden ist daher überflüssig. In der nächsten Admin Mod Version hätte es eigentlich entfernt werden sollen. Mangels einer neueren Version, ist es immer noch in der aktuellen enthalten.

Den Eintrag in der plugin.ini und das Plugin einfach entfernen.

5.12. plugin_TFC

Das Team Fortress Classic (TFC) Plugin erlaubt es, Spieler in andere Teams zu transferieren, die Variable „tfc_clanbattle_prematch“ zu verändern, und es enthält eine rudimentäre Teambalancing-Funktion.

5.12.1. admin_balance

`admin_balance <0/1>`

Mit diesem Befehl kann man das Teambalancing ein- (1) und ausschalten (0).

Beispiel:
`admin_balance 1`

Hiermit wird das Teambalancing eingeschaltet.

Access-Level: 32
Gehört zu: plugin_TFC

5.12.2. admin_blue

`admin_blue <ID/IP/Name>`

Man kann mit diesem Befehl einen Spieler ins blaue Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben.

Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiele:

```
admin_blue "Blaues Veilchen"
admin_blue Big
admin_blue STEAM_0:123456
admin_blue 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Blaues Veilchen“ ins blaue Team verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ins blaue Team verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_TFC

Siehe auch:
admin_green, admin_red, admin_yellow, allow_client_exec

5.12.3. admin_green

`admin_green <ID/IP/Name>`

Man kann mit diesem Befehl einen Spieler ins grüne Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben.

Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiele:

```
admin_green "Gruenes Gras"
admin_green Big
admin_green STEAM_0:123456
admin_green 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Gruenes Gras“ ins grüne Team

verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ins grüne Team verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_TFC

Siehe auch:
admin_blue, admin_red, admin_yellow, allow_client_exec

5.12.4. admin_prematch

`admin_prematch <Minuten>`

Der Befehl admin_prematch erlaubt es, die Zeit der Aufwärmrunde vor einem Clanwar zu definieren.

Beispiel:
`admin_prematch 5`

Das Beispiel zeigt, wie man die Zeit der Aufwärmrunde auf 5 Minuten stellt.

Access-Level: 4
Gehört zu: plugin_TFC

5.12.5. admin_red

`admin_red <ID/IP/Name>`

Man kann mit diesem Befehl einen Spieler ins rote Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben.

Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiele:
```
admin_red "Roter Klatschmohn"
admin_red Big
admin_red STEAM_0:123456
admin_red 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Roter Klatschmohn“ ins rote Team verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im

Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ins rote Team verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_TFC

Siehe auch:
admin_blue, admin_green, admin_yellow, allow_client_exec

5.12.6. admin_yellow

```
admin_yellow <ID/IP/Name>
```

Man kann mit diesem Befehl einen Spieler ins gelbe Team verschieben. Ist er nicht tot, wird er bei dieser Aktion sterben.

Der Spieler kann entweder über seine ID, IP, den Namen oder einen eindeutigen Namensteil erkannt werden.

Damit der Befehl funktioniert, muss allow_client_exec aktiviert worden sein.

Beispiele:

```
admin_yellow "Gelber Loewenzahn"
admin_yellow Big
admin_yellow STEAM_0:123456
admin_yellow 23.156.43.12
```

Im ersten Beispiel wird der Spieler mit dem Namen „Gelber Loewenzahn“ ins gelbe Team verschoben. Im nächsten Beispiel wird der Spieler mit dem Namen „Big“ oder „Big“ im Namen (sofern kein anderer Spieler „Big“ im Namen trägt) ins gelbe Team verschoben. Dies funktioniert auch mit einer ID oder IP wie in den letzten beiden Beispielen.

Access-Level: 8192
Gehört zu: plugin_TFC

Siehe auch:
admin_blue, admin_green, admin_red, allow_client_exec

6. FAQ

6.1. admin_restartround, admin_ct oder admin_t funktionieren nicht!

admin_restartround, admin_ct und admin_t sind Befehle aus dem Standardplugin plugin_CS. Dieses ist bei der Installation von Admin Mod nicht automatisch aktiv, da Admin Mod auch für andere Mods neben Counter-Strike eingesetzt werden kann.

Es muss die Raute „#“ vor dem Eintrag in der plugin.ini entfernt werden. Mehr dazu im Kapitel 4.2.

6.2. Admin Mod funktioniert nicht!

Das liegt meistens an einer fehlerhaften Installation. Im Weiteren ist ein Checkliste aufgeführt, mit der man Schritt für Schritt die wesentlichen Fehlerquellen ausmerzen kann.

1. Wurde die Installationsanleitung nochmal Schritt für Schritt durchgegangen?
2. Funktioniert Metamod? Der Befehl „meta version“ zeigt, welche Metamod-Version läuft. Wird nichts angezeigt, ist vermutlich die liblist.gam nicht editiert oder der Server wurde nicht neu gestartet.
3. Funktioniert Admin Mod? Befehl „meta list“ und „admin_cmd admin_version“ geben in der Serverconsole darüber Aufschluss.
4. Wird ein Dedicated Server unter Steam (nur die Version über das Menü „Games“) verwendet, wird die liblist.gam von Steam überschrieben, daher liblist.gam schreibschützen oder wahlweise den -dll Startparameter verwenden.
5. Wenn der Server gar nicht startet, sollte überprüft werden, ob in server.cfg (bzw. listenserver.cfg) der Eintrag „exec addons/adminmod/config/adminmod.cfg“ gesetzt ist.
6. Gleiches kann auch passieren, wenn das „password_field“ ohne Unterstrich am Anfang geschrieben wurde, z.B. „pw-home“ statt richtig „_pw-home“.
7. Wichtig ist außerdem, dass die Groß- und Kleinschreibung unter Linux beachtet wird. Es macht einen deutlichen Unterschied, ob man admin_MM_i386.so oder

admin_mm_i386.so in der plugins.ini von Metamod schreibt.

6.3. Wo bekomme ich weitere Plugins her?

Um die Fähigkeiten Admin Mods über die der Standardplugins zu erweitern, muss man sich sogenannte Custom Plugins besorgen. Eine Vielzahl von Plugins ist auf diversen Seiten im Netz zu finden.

- Deutsche Admin Mod Webseite[1]
- Englische Admin Mod Webseite[2]
- Deutsches Admin Mod Forum[3]
- Webseite von Ravenous Bugblatter[4]

6.4. Admin Mod Befehle funktionieren nicht (z.B. in HLSW)

Eine kurze Beschreibung des Unterschieds zwischen Client-Console und Server-Console kann an dieser Stelle weiterhelfen.

Wenn man online auf einem Server spielt, und in die Console geht, befindet man sich in der Client-Console. Admin Mod Befehle werden hier so eingegeben, wie das bekannt ist, also beispielsweise `admin_help`. Serverbefehle müssen stets mit „rcon“ vorweg geschrieben werden. Dadurch weiß der Server, dass man einen Befehl in die Serverconsole schreiben will.

Einen direkten Zugriff auf die Serverconsole hat man hingegen im Fenster des Dedicated Servers oder per RCon-Console bei einem Programm (z.B. HLSW[5]). Hier muss kein „rcon“ vor die Serverbefehle gesetzt werden. Man ist bereits in der Serverconsole. Allerdings müssen nun „admin_cmd“ oder wahlweise „admin_command“ vor die Admin Mod Befehle gestellt werden, damit der Server sie als solche erkennt (z.B. `admin_cmd admin_help`).

6.5. Couldn’t find Meta_Query()

Die folgende Fehlermeldung tritt auf:

[1] http://www.adminmod.de
[2] http://www.adminmod.org
[3] http://forum.adminmod.de
[4] http://www.ravenousbugblatterbeast.pwp.blueyonder.co.uk
[5] http://www.hlsw.de

```
'<amx_admin.so>'; Couldn't find Meta_Query()
```

Irgendwo im Netz hat mal jemand behauptet, dass die amx_admin.so auch in die plugins.ini von Metamod kommt. Offensichtlich hat diese Information die „Runde“ gemacht. Dies Behauptung ist selbstverständlich Unsinn.

Es handelt sich um die Scripting-Engine. Einzige Vorraussetzung ist, dass sie sich im gleichen Verzeichnis wie die admin_MM_i386.so befindet.

Gleiches gilt natürlich auch für die Windows-Version.

6.6. WARNING: meta-interface version mismatch

Es ist möglich, dass evtl. Folgendes in den Logdateien auftaucht:

```
09:58:53 [LOGD] WARNING: meta-interface version mismatch;
requested=5:9 ours=5:8
09:58:53 [LOGD] WARNING: metamod version is newer than expected;
consider finding a newer version of this plugin
09:58:53 [META] dll: Note: Plugin ' ' interface version didn't match;
expected 5:9, found 5:8
09:58:53 [META] dll: Note: plugin ' ' is not using the latest interface
version; there may be an updated version of the plugin
09:58:53 [LOGD] LogDaemon v1.00.5, 2002/09/17
09:58:53 [LOGD] by Will Day, Tim Thelin, Pieter de Zwart, David Mart
09:58:53 [LOGD] compiled: Sep 17 2002, 22:59:31 PST (optimized)
```

Es handelt sich dabei nur um eine Warnung. In diesem Fall ist LogD aber immer noch kompatibel, auch wenn es nicht mit der neusten Metamod SDK compiliert wurde. Die Meldung kann daher ignoriert werden.

7. Erweiterungen

7.1. LogDaemon (LogD)

Einige Plugins benötigen zusätzlich noch LogD (LogDaemon), das auf der LogD-Seite[1] bzw. die 64bit-Version auf Adminmod.de[2] zu bekommen ist. LogD ist ein Metamod-Plugin, das die Logeinträge von Half-Life in Realtime parsed und damit die Möglichkeiten von Adminmod erweitert (z.B. Teamkillerkennung in Counter-Strike). Es wurde zwar zwischenzeitlich an erweiterten Funktionen in Adminmod selber gearbeitet, was aber nie über eine leicht verbuggte Betaversion hinaus geführt hat.

7.1.1. Installation

Linux:
`Erstelle Verzeichnis addons/logd/dlls`
`Kopiere logd_mm_i386.so ⟶ addons/logd/dlls`

Windows:
`Erstelle Verzeichnis addons\logd\dlls`
`Kopiere logd_mm.dll ⟶ addons\logd\dlls`

Anschließend muss LogD noch in der addons/metamod/plugins.ini durch folgende Einträge aktiviert werden.

```
linux addons/logd/dlls/logd_mm_i386.so
win32 addons/logd/dlls/logd_mm.dll
```

Nun muss nur der Gameserver neugestartet werden. Ob LogD funktioniert, kann man über die Serverconsole oder per RCon prüfen. Dazu schreibt man “meta list”. Als Ergebnis sollte Folgendes in etwa zu sehen sein:

```
[x]LogDaemon RUN - logd_mm.dll v1.00.6 ini ANY Pause
```

Wichtig ist dabei der Begriff “RUN”.

Eine weitergehende Konfiguration ist in der Regel nicht mehr notwendig. Ggf. muss die Variable mp_logdetail editiert oder auch logd_block gesetzt werden. Die Einstellung für mp_logdetail wird einem in der Regel in der Readme des jeweiligen Plugins mitgeteilt.

[1] http://logd.sf.net/
[2] http://www.adminmod.de/

Der Befehl logd_block ist optional (s. Abschnitt 7.1.4). Die Plugins sind nichts anderes als Admin Mod Plugins mit erweiterter Funktionalität und werden daher genauso wie alle anderen installiert.

7.1.2. Variablen

7.1.2.1. logd_debug

`logd_debug <#>`

Setzt man hier den Wert ungleich 0, werden einige Informationen zum Parsingvorgang in die Logdateien geschrieben. LogD ist leider nicht mit allen Mods kompatibel, da sich einige Programmierer nicht an die Vorgaben von Valve für die Log-Syntax halten. Mit der Debugfunktion kann schnell erkannt werden, was LogD nicht erkennt. In der Regel setzt man diese Variable aber nicht.

7.1.2.2. logd_version

`logd_version <#>`

Diese Variable zeigt ausschließlich die LogD-Version an. Ursprünglich wollte Wraith, diese nie in LogD einbauen, da es von außen unerheblich zu erkennen ist, in welcher Version LogD läuft. Letzlich hat er sich aber überreden lassen, da die Admin Mod Scripter gerne in ihren Plugins überprüfen wollten, ob LogD installiert ist. Die Variable wird automatisch von LogD gesetzt.

7.1.3. Befehle

7.1.3.1. logd_block

`logd_block <#>`

Mit diesem Befehl kann man verhindern, dass bestimmte Ereignisse in die Logdateien geschrieben werden. Damit kann man die Größe der Logdateien teilweise massiv reduzieren. Dies kann insbesondere bei der Auswertung von Treffern sinnvoll sein (mp_logdetail > 0). In Abschnitt 7.1.4 wird darauf näher eingegangen.

Beispiel:
`logd_block 58`

In diesem Beispiel wird verhindert, dass Treffer in die Logdateien geschrieben werden.

7.1.3.2. logd_reg

`logd_reg <#> <Befehl>`

Dieser Befehl registriert einen Server-Befehl, den LogD bei einem bestimmten Ereignis (Nummer) ausführt und an den die zugehörigen Parameter weitergegeben werden. Diesen Befehl setzt in der Regel ein Admin Mod Plugin ab. Man kann ihn auch per Hand setzen, was aber in den meisten Fällen nicht sinnvoll ist.

Beispiele:

```
logd_reg 57 admin_command logd_kill
logd_reg 51 say Neuer Spieler!
logd_reg 62 admin_command admin_tsay Rundenstart oder Rundenende
```

Im ersten Beispiel wird der Admin Mod Befehl „logd_kill“ für das Ereignis 57 (Kill) registriert. Der Befehl wird bei jedem Kill ausgeführt. Dies ist ein typischer Aufruf in einem Plugin.

Das zweite Beispiel zeigt, wie man LogD auch ohne Admin Mod nutzen kann. Hierbei wird bei Ereignis 51 (Spieler tritt dem Spiel bei) die Nachricht „Neuer Spieler!“ in den Chat geschrieben. Allerdings wird der Userindex noch dahinter angezeigt.

Im letzten Beispiel wird beim Ereignis 62 (allgemeine Ereignisse) die Nachricht „Rundenstart und Rundenende“ links auf dem Bildschirm gezeigt. Dazu wird aber auch „Round_End“ oder „Round_Start“ angezeigt.

Wie man bei den letzten beiden Beispielen sehen kann, ist Admin Mod nicht wirklich für LogD notwendig, aber man kann die Parameter nicht auswerten bzw. muss sie mit anzeigen.

Außerdem ist wichtig zu bemerken, dass beim Mapstart alle Registrierungen zurückgesetzt werden.

7.1.3.3. logd_reginfo

`logd_reginfo`

Dieser Befehl gibt aus, welche Befehle zu welchen Ereignissen registriert wurden.

Beispiel:

```
LogD callback registration information:
Event Name          Event Code   Function to call
Enter               51           admin_command im_connect
Disconnect          52           admin_command im_disconnect
Suicide             53           admin_command im_suicide
Team Selection      54           admin_command allowsound_team
```

```
Kill                57          admin_command logd_teamkill
Kill                57          admin_command im_kill
Kill                57          admin_command hp_kill
Injury              58          admin_command im_injure
Injury              58          admin_command hp_injure
Player Action       60          admin_command im_playeraction
Player Action       60          admin_command hp_paction
Team Action         61          admin_command im_teamaction
Team Action         61          admin_command hp_taction
World Action        62          admin_command im_world
World Action        62          admin_command hp_reset
```

So sieht eine typische Rückmeldung auf „logd_reginfo“ aus.

7.1.3.4. logd_status

`logd_status`

Dieser Befehl gibt aus, welche Ereignisse registriert wurden und ob sie aus den Logdateien gefiltert werden.

Beispiel:

```
LogD parser status:
Event Name          Event Code   Active?  Blocking?
Cvar                1            No       No
Log                 2            No       No
Map                 3            No       No
Rcon                4            No       No
Server Name         5            No       No
Server Chat         6            No       No
Connect             50           No       No
Enter               51           Yes      No
Disconnect          52           Yes      No
Suicide             53           Yes      No
Team Selection      54           Yes      No
Role Selection      55           No       No
Name                56           No       No
Kill                57           Yes      No
Injury              58           Yes      No
PvP Action          59           No       No
Player Action       60           Yes      No
Team Action         61           Yes      No
```

```
World Action        62            Yes       No
Player Chat         63            No        No
Team Alliance.      64            No        No
Team Score          65            No        No
Private Chat        66            No        No
```

So sieht eine typische Rückmeldung auf „logd_status“ aus.

7.1.4. Performance und Plattenplatz

Es kommt immer wieder das Gerücht auf, dass die Verwendung LogD zu Einbrüchen in der Serverperformance führe.

Grundsätzlich gilt natürlich, dass mit jedem installiertem Metamod-Plugin der Server mehr belastet wird. LogD jedoch für Performanceprobleme verantwortlich zu machen, ist in der Regel eine Mär. Die LogD-Programmierer hatten den Fehler gemacht auf ihrer Seite Performanceprobleme nicht auszuschließen. Auch Admin Mod oder andere Plugins können Probleme bereiten, was meist auf eine Fehlkonfiguration oder ineffiziente Plugins zurückzuführen ist.

Für viele Gameserveranbieter war es immer wieder ein guter Vorwand eigene Probleme auf den Benutzer abzuwälzen.

Nutzt man TK-Plugins (TK = Teamkill) kann es aber durchaus bei sehr schwachen Servern zu Performance-Problemen kommen, weil man mp_logdetail ungleich 0 laufen lassen muss, um alle Arten von Treffern zu loggen. Die um das zehnfache vergrößerten Logdateien können dabei zu Lags führen, aber diese Einträge lassen sich auch mit LogD herausfiltern. Das spart Plattenplatz und reduziert die Schreibvorgänge, das Problem rührt aber nicht von Seiten LogD her.

LogD hat zum Filtern den Befehl logd_block <event>. Der zugehörige Event für die Treffer ist 58. Der Eintrag in der server.cfg muss also lauten:

```
logd_block 58
```

Dabei ist aber zu beachten, dass dann externe Statistikprogramme (z.B. Psychostats[3]) diese Daten nicht mehr auswerten können.

Schaut man sich das Alter von LogD an, so ist nachvollziehbar dass LogD selber keine Performanceprobleme verursacht. Heutzutage haben Server eine vielfach höhere Leistung als damals. LogD fällt da nicht mehr auf.

[3] http://www.psychostats.com/

7.2. StatsMe

Einige Admin Mod Plugins benötigen über Admin Mod selber hinaus auch Statsme[4]. Statsme ist wie LogD ein Metamod Plugin, welches diverse Events zur Verfügung stellt, die man mit Admin Mod abgreifen und auswerten kann. Dabei geht Statsme vielfach über das hinaus, was LogD kann, da es nicht die Logs liest, sondern alle Events auslesen kann, die die Half-Life Engine bietet.

Statsme wurde von Admin Mod Seite recht stiefmütterlich behandelt, da es gerne zur „Verschönerung“ des Servers verwendet wurde. Es war laut und bunt. Man kann aber Statsme auch recht geräuschlos nebenbei laufen lassen. Einige recht gute Customplugins sind dadurch entstanden.

7.2.1. Installation und Konfiguration

Die Dateien liegen bereits in ihrer Struktur fertig für die Installation vor.

```
Kopiere Verzeichnis addons
```

Anschließend muss Statsme noch in der addons/metamod/plugins.ini durch folgende Einträge aktiviert werden.

```
linux addons/statsme/dlls/statsme_mm_i386.so
win32 addons/statsme/dlls/statsme_mm.dll
```

Nun ist nur der Gameserver neu zu starten. Ob Statsme funktioniert, kann man über die Serverconsole oder per RCon prüfen. Dazu schreibt man “meta list”. Als Ergebnis sollte Folgendes in etwa zu sehen sein:

```
[x]STATSME RUN - statsme_mm.dll v2.8.3 ini Chlvl Chlvl
```

Wichtig ist dabei der Begriff “RUN”. Weitere Einstellungen müssen nicht getätigt werden.

7.2.2. Performance und Plattenplatz

Wie LogD braucht Statsme etwas Rechenleistung. Da im Hintergrund deutlich mehr verarbeitet wird, dürfte Statsme auch etwas mehr Rechenleistung beanspruchen als LogD. Bei einem sehr leistungsschwachen Server kann sich das bemerkbar machen, aber solche Rechner setzt heute niemand mehr ein.

Durch die Zusatzeinträge in den Logdateien ist es erste Wahl z.B. für Psychostats[5], geht aber damit nicht weniger sparsam mit dem Plattenplatz um als LogD. Ggf. sollte man darüber nachdenken, ob man das Logging nicht abschaltet, sofern man auf externe

[4] http://www.unitedadmins.com/index.php?p=content&content=statsme
[5] http://www.psychostats.com/

Statistikprogramme verzichten will oder kann.

7.3. HLStats

Bei HLStats[6] handelt es sich um ein Perl-Skript, das in Echtzeit die Logs auswertet. Die Ergebnisse werden in eine MySQL-Datenbank geschrieben und sind damit über eine Webseite abrufbar.

Ab der HLStats-Version 1.40 ist das sogenannte Skill-Pack schon inklusive. Dieses sorgt dafür, dass man nach Chat-Eingabe von „/hls_skill“ bzw. „/hls_pskill“ seinen Rang und die erreichten Skill-Punkte angezeigt bekommt. Letzteres zeigt nur dem aufrufenden Spieler selbst diese Information. Eine Meldung direkt an einen einzelnen Spieler erfordert jedoch ein Metamod-Plugin wie Admin Mod.

Um die Funktion „/hls_pskill“ zu aktivieren sind folgende Einträge in der hlstats.conf zu machen:

```
Rcon 1
RconSay "admin_psay"
```

Außerdem sollte man im Webinterface nicht vergessen, das Rcon-Passwort unter „Game Settings/<mod>/Servers“ einzugeben. Ansonsten könnte es passieren, dass der HLStats-Server vom Gameserver ausgesperrt wird.

7.4. Metamod Plugins

Neben LogD und StatsMe gibt es noch weitere Metamod-Plugins, die bestimmte Server-Funktionen nachrüsten. Auch wenn diese keine definierte Schnittstelle zu Admin Mod besitzen, können sie dennoch von Admin Mod ferngesteuert werden. Vielfach gibt es sogar Admin Mod Plugins, die dafür entwickelt wurden, die Administration zu verbessern, bzw. das Rechtelevelsystem für diese Metamod-Plugins nutzbar zu machen. Für die Metamod-Plugin spezifischen Serverbefehle werden dabei Aliase in Form von Admin Mod-Befehlen erstellt.

Eine, wenn auch nicht vollständige, Liste der Metamod-Plugins lässt sich auf der Metamod-Webseite[7] einsehen. Einige Admin Mod Plugins zu ihrer Administration sind auf Adminmod.de[8] oder Adminmod.org[9] im Downloadbereich zu finden.

[6] http://www.hlstats-community.org
[7] http://metamod.org/plugins.html
[8] http://www.adminmod.de
[9] http://www.adminmod.org

8. Scripting

Das Scripting-Kapitel ist in Form eines Tutoriums gefasst, bei dem nicht bei Adam und Eva angefangen wird. Auch wird das Thema nicht in allen Feinheiten vorgestellt. Das würde den Sinn eines Tutoriums verfehlen. Ich bin leider nur Autodidakt und daher im Fachvokabular eines Informatikers nicht firm. Letztendlich sollte mein Wissen jedoch ausreichen, Laien das Admin Mod Scripting näher zu bringen.

Für das Schreiben von Plugins (oder Skripts) unter Admin Mod empfiehlt es sich, bereits Grundkenntnisse in einer Programmiersprache zu besitzen. Dabei genügen bereits Kenntnisse in Basic oder PHP, C++ Kenntnisse wären ideal. Das Wissen, wie ein HL-Server funktioniert, ist jedoch die wichtigste Voraussetzung!

Die verwendete Skriptsprache heißt Small (inzwischen Pawn genannt)[1] und bietet gegenüber C++ nur eingeschränkte Möglichkeiten der Programmierung. Gerade deshalb wurde sie gewählt, da der Anwender maximal in der Lage ist, den HL-Server abstürzen zu lassen, nicht aber das gesamte Betriebssystem.

Zunächst sollen die Konventionen und der Syntax von Small vorgestellt werden. C++ Anhänger werden sich vermutlich wie zu Hause fühlen.

8.1. Syntax

Die Syntax Smalls lässt sich am Besten anhand eines „Hallo Welt“ Codes erklären:

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

new VERSION[]= "1.0";

public plugin_init() {
   plugin_registerinfo("Hallo Welt Plugin","Hallo Welt!",VERSION);
   say("Hallo Welt!");

   return PLUGIN_CONTINUE;
}
```

[1] http://www.compuphase.com/pawn/pawn.htm

Die Funktion plugin_init wird beim Starten des Plugins ausgeführt (entspricht der main() Funktion von C++). Die Funktion muss öffentlich bekannt sein (public) und sollte PLUGIN_CONTINUE zurückgeben. Der Beginn und das Ende von Verzweigungen, Schleifen, usw. wird durch geschweifte Klammern kenntlich gemacht. Funktionen müssen durch ein Semikolon abgeschlossen werden. Includes und Deklarationen werden mittels Hashes (#) kenntlich gemacht. Außerdem ist zu beachten, dass Small Groß- und Kleinschreibung unterscheidet.

8.2. Datentypen, Variablen und Konstanten

Small wird als „typenlose“ Programmiersprache bezeichnet. Der einzige, direkt unterstützte Datentyp ist 32-bit signed Integer (-2147483648 bis +2147483647).

Variablen werden in der Regel über die „new“-Anweisung deklariert. Man kann sie „global“ definieren, wenn sie im Skript außerhalb der Funktionen deklariert werden oder „lokal“ innerhalb ein Funktion.

```
new iZahl1;
new iZahl2 = 1;
```

Man kann die Variable ohne Zuweisung erstellen und diese im Code nachholen oder, wie im zweiten Beispiel gezeigt, direkt zuweisen.

Lokale Variablen gelten nur für die Funktion, in der sie erstellt wurden und werden nach Abschluss der Ausführung selbiger wieder gelöscht. Daher findet man hin und wieder auch statt „new“ „static“ bei der Deklarierung, da hier der Variableninhalt erhalten bleibt. Eine andere Möglichkeit den Variableninhalt zu behalten und ihn zudem allen Funktionen gleichzeitig zur Verfügung zu stellen, besteht in der Definition globaler Variablen. Dabei muss die Variable außerhalb der Funktionen definiert werden. Meist findet man diese am Kopf des Plugins wieder, damit sie den nachfolgenden Funktionen bekannt sind.

Zahlen können auch in bis zu zweidimensionalen Feldern verwendet werden.

```
new aFeld[2][2];
aFeld[1][0] = 5;
aFeld[1][1] = 3;
```

Eine einmal festgelegte Feldgröße kann während der Laufzeit nicht geändert werden. Man sollte vorausschauend programmieren und die Feldgröße ausreichend dimensionieren.

Die 1 und die 0 sieht Small neben den üblichen Rechnungen insbesondere in Vergleichen auch als Boolean-Wert „Wahr“ bzw. „Falsch“ an.

Konstanten werden stets am Kopf des Plugins definiert. Die Definition sieht etwa folgendermaßen aus:

```
#define MAX_TEST_LENGTH 100
```

Dies ist eine compilerspezifische Anweisung und muss damit nicht mit einem Semikolon abgeschlossen werden. Der Compiler ersetzt im Beispiel alle `MAX_TEST_LENGTH` beim Compilieren durch 100. Eine Feldvariable kann also auch so definiert werden:

```
new Variable[MAX_TEST_LENGTH];
```

Das mag auf den ersten Blick etwas umständlich erscheinen. Hat man jedoch mehrere dieser Felder, kann man die Größe aller bequem über die Konstantendefinition ändern. Darüber hinaus erhöht es die Lesbarkeit des Programmcodes.

Alternativ kann man Konstanten auch mit einem vorangestellten „const" definieren:

```
const cVariable = 1
```

Strings werden in Feldern abgespeichert, wobei jeder Buchstabe eine Zahl in einer Zelle des Arrays ist. Das Feld zum zugehörigen Text muss immer um eine Zelle größer als die Länge des Texts angesetzt werden, da Small in die letzte Zelle ein „^0" setzt, um das Ende des Strings festzulegen.

```
new Text[200]="Hallo Welt!";
new Text[]="Hallo Welt!";
```

Man kann wie gezeigt auch auf die Deklaration einer Feldgröße verzichten. Die Feldgröße beträgt in diesem Fall jedoch nur die Länge von „Hallo Welt!" + 1 (12) und kann auch nicht während der Laufzeit vergrößert werden.

Neben dem Pseudo-Datentyp String gibt es den Typ der Festkommazahlen (Fixed). Die Festkommazahlen haben in Admin Mod einen Zahlenbereich von -2147482 bis +2147482 mit 3 Nachkommastellen. Festkommazahlen werden folgendermaßen deklariert:

```
new fixed:fPI = 3.142;
```

Die Information „fixed:" weist den Compiler an, die Zahl als Festkommazahl zu betrachten.

Zu guter Letzt kann man auch Enums definieren. Dabei werden Begriffe einer aufsteigenden Zahlenreihe zugeordnet. Man kann sie zur besseren Lesbarkeit des Codes einsetzen.

```
enum tks{tk=0,victim};
```

Das „=0" bedeutet, dass bei Null angefangen wird zu zählen. Jeder weitere Option bekommt einen um eins erhöhten Wert.

8.3. Operatoren

Es stehen folgende Rechenoperatoren zur Verfügung:

`+` Addieren
`-` Subtrahieren
`*` Multiplizieren

`/` Dividieren
`&` Bit-Shift-Aktionen

Auch existieren die üblichen Vereinfachungen aus anderen Sprachen.

`i++;` entspricht `i=i+1;`
`i+=j;` entspricht `i=i+j;`

Vorsicht! Nicht auf Zellen von Feldern anwenden. `a[1]++;` z.B. führt auf Grund eines Compilerbugs im schlimmsten Fall zum Absturz des HL-Servers!

Eine Besonderheit sind die Festkommazahlen. Während das Addieren und Subtrahieren noch identisch ausgeführt wird, gibt es für das Multiplizieren und Dividieren spezielle Funktionen (fmul und fdiv). Über die Grundrechenarten hinausgehende Rechenfunktionen bzw. -operatoren zu Festkommazahlen sind der math.inc zu entnehmen.

Es gibt darüber hinaus auch logische Operatoren, die insbesondere in Verzweigungen eingesetzt werden:

`!` Nicht
`&&` Und
|| Oder

8.4. Verzweigungen

Es gibt in Small zwei Verzweigungen, „if“ und „switch“. Die If-Verzweigung kann durch „else if“ und/oder „else“ erweitert werden.

```
vergleich(i){
   new j=0;

   if (i = 1){
      j = 2;
   }
   else if (i = 2 || i = 3) {
      j = 5;
   }
   else {
      j = 1;
   }

   return j;
}
```

Die Vergleiche wurde zur besseren Lesbarkeit in runde Klammern gesetzt. Dies ist nicht notwendig, aber zu empfehlen. Für Konstrukte dieser Art sollte man aus Übersichtlichkeitsgründen aber eher auf die Switch-Verzweigung ausweichen. Das obige Beispiel sieht dann folgendermaßen aus:

```
vergleich(i){
```

```
    new j = 0;

    switch(i){
       case 1:{
          j = 2;
       }
       case 2,3:{
          j = 5;
       }
       default:{
          j = 1;
       }
    }

    return j;
}
```

Switch-Verzweigungen können nur Zahlen jedoch keine Strings vergleichen.

8.5. Schleifen

Small hat 3 verschiedene Schleifen implementiert, „for“, „while“ und „do ... while“. In Admin Mod werden vorzugsweise For-Schleifen eingesetzt, da auf Grund der festgelegten Feldgröße auch die Schleifenlänge bereits vorab festgelegt werden kann.

```
schleife(){
   new j = 0;

   for (i=0;i<=5;i++){
      j = j + i;
   }

   return j
}
```

In diesem Fall läuft die Schleife von 0 bis 5, wobei i um eins bei jedem Schritt erhöht wird. Will man die Schleife abbrechen, z.B. wenn j 10 überschreitet, muss man mit der break-Anweisung arbeiten.

```
schleife(){
   new j = 0;

   for (i=0;i<=5;i++){
      if (j >= 10){
         break;
      }
      j = j + i;
   }

```

```
    return j
}
```

In diesem Fall bricht die For-Schleife komplett ab, und der Wert j wird zurückgegeben. Statt „break“ kann auch „continue“ verwendet werden.

```
            continue;
```

Mit „continue“ wird direkt der nächste Durchlauf der Schleife initiiert, ohne dass die noch ausstehenden Rechnungen (hier: `j = j + i;`) durchgeführt werden.

Die gleiche Schleife kann auch mit „while“ und „do ... while“ durchgeführt werden, wobei beachtet werden muss, dass „do ... while“ mindestens einmal ausgeführt wird, da der Vergleich erst am Ende der Schleife durchgeführt wird.

```
schleife(){
   new j = 0;

   do i++ while (j >= 10){
      j = j + i;
   }

   return j
}
```

Und hier auch ein Beispiel mit while:

```
schleife(){
   new j = 0;

   while (j >= 10) i++{
      j = j + i;
   }

   return j
}
```

8.6. Direktiven

Direktiven sind Anweisungen an den Compiler, die nur zur Laufzeit des Compilers und nicht während der Laufzeit des Plugins berücksichtigt werden. Es sollen an dieser Stelle nur die gebräuchlisten Direktiven in Admin Mod beschrieben werden.

8.6.1. include

#include Include-Datei oder <Include-Datei>

Mit der Include-Direktive wird eine Include-Datei mit seinen Funktionen in den Quellcode eingebunden. Der Dateiname muss mit „.inc“ enden. Diese Endung darf jedoch nicht in der Direktive auftauchen. Ist der Dateiname in Klammern, wird die Include-Datei im Includes-Verzeichnis gesucht, während sie ohne die Klammern im gleichen Verzeichnis wie die Quelldatei erwartet wird.

8.6.2. define

`#define KONSTANTENNAME Zahl`

Mit dieser Direktive werden, wie bereits im Kapitel „Datentypen, Variablen und Konstanten“ beschrieben, Konstanten festgelegt.

8.6.3. if, elseif, else und endif

```
#define TEST 1
#if Test == 1
   Small-Code 1
#elseif Test == 2
   Small-Code 2
#else
   Small-Code 3
#endif
```

Man kann damit über eine Konstante definieren, welchen Code der Compiler für das Plugin nutzen soll. Ich würde das als Poor-Man’s-Option bezeichnen. Für den Enduser sollte man die Option üblicherweise zur Laufzeit änderbar machen (z.B. mit einem Admin Mod Befehl).

8.7. Funktionen und Events

Funktionen sind ein wesentlicher Bestandteil Admin Mods. Funktionen können aus anderen (einfacheren) Funktionen erstellt werden. Viele Funktionen werden jedoch auch von Admin Mod zur Verfügung gestellt. Sie werden in den sogenannten Includes vermerkt und stehen nach Einbindung selbiger für die Programmierung zur Verfügung.

Es gibt aber auch Funktionen, die bei bestimmten Events (z.B. Serverbefehl oder Spielerconnect) automatisch ausgeführt werden. Dabei kann die zugehörige Reaktion programmiert werden. Dies sind:

- plugin_init (Serverstart bzw. nach Mapwechsel, vergleichbar z.B. in C mit main())
- plugin_connect (Spielerconnect)
- plugin_info (Spieleraktionen beim Connect, aber auch Namen- oder Modelwechsel)

- plugin_disconnect (Spielerdisconnect)
- plugin_command (Serverbefehle)
- abgelaufener Timer
- abgelaufener Vote

Auf die Anwendung von Events wird im Tutorial näher eingegangen.

Vorgegebenen Funktionen ist ein „stock“ (in Small erstellte Funktion) bzw. „native“ (von Admin Mod zur Verfügung gestellte Funktion) vorangestellt. Sie werden nur vom Compiler eingebunden, wenn sie auch benötigt werden. Beide Schlüsselwörter sind nur von Interesse, wenn man eigene Includes erstellen möchte. Funktionen, die auf Events reagieren, muss ein „public“ vorangestellt werden, da anderenfalls die Funktion zur Laufzeit nicht gefunden wird. Alle weiteren Funktionen benötigen keine besondere Kennzeichnung.

8.7.1. Beispiele für Events von Admin Mod

Zunächst startet Metamod Admin Mod. Admin Mod lädt seine Plugins in der Reihenfolge, wie sie in der plugin.ini stehen und führt jeweils die plugin_init Funktion aus. Die plugin_init sollte so programmiert sein, dass Pluginname, Beschreibung, Version und ggf. zu registrierende Befehle an Admin Mod zurückgegeben werden. Admin Mod weiß nun, welche Befehle es an das Plugin weiterleiten soll.

Kommt ein Spieler auf den Server, ruft Admin Mod die Funktion plugin_connect auf (sofern im Plugin definiert) und führt die angegebenen Aktionen durch.

Wird ein im Plugin definierter Befehl aufgerufen, wird die zugehörige Funktion aufgerufen, die (sofern der Accesslevel des Spielers ausreicht) ausgeführt wird. Das Plugin meldet zurück, ob Admin Mod noch andere Plugins abfragen soll (return PLUGIN_CONTINUE), oder ob Admin Mod die weiteren Plugins übergehen soll (return PLUGIN_HANDLED).

8.8. Tutorial

In diesem Tutorial wird auf das Grundgerüst aufgebaut und die wichtigsten Funktionen und Events sowie deren Anwendung vorgestellt.

8.8.1. Basis

Small bzw. Admin Mod erwarten ein gewisses Grundgerüst, damit sie das compilierte Plugin akzeptieren.

```
1   #include <core>
2   #include <string>
3   #include <admin>
4   #include <adminlib>
5
6   public plugin_init() {
7      plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
8      return PLUGIN_CONTINUE;
9   }
```

Zunächst werden die Includes eingebunden.

```
1   #include <core>
2   #include <string>
3   #include <admin>
4   #include <adminlib>
```

Man sollte nie zu sparsam bei den Includes sein. Theoretisch kann man alle vorhandenen Includes einbinden, da nur das vom Compiler berücksichtigt wird, was er auch benötigt. Das Plugin wird durch mehr Includes nicht größer.

Damit das Plugin überhaupt etwas tut, wird das Event plugin_init eingebunden. Überlicherweise geschieht dies am Ende des Quellcodes.

```
6   public plugin_init() {
7      plugin_registerinfo("Testplugin","Das Plugin macht noch nichts!","1.0");
8      return PLUGIN_CONTINUE;
9   }
```

Das Event plugin_init wird nach dem HL-Serverstart bzw. nach jedem Mapwechsel von Admin Mod aufgerufen. Damit Admin Mod erkennt, dass das Plugin plugin_init implementiert hat, muss das Event mit „public“ bekanntgegeben werden. In Zeile 8 wird Admin Mod zurückgemeldet, dass die Abarbeitung des Events erfolgreich beendet ist.

In Zeile 7 wird Admin Mod mitgeteilt (plugin_registerinfo), wie das Plugin heißt, was das Plugin macht, und um welche Version des Plugins es sich handelt.

```
7      plugin_registerinfo("Testplugin","Das Plugin macht noch nichts!","1.0");
```

Die angebenen Informationen haben keine Funktion in Admin Mod, sondern dienen nur zur späteren Darstellung in der Console, wenn man sich alle geladenen Plugins auflisten lässt.

Dieses Plugin registriert sich also lediglich bei Admin Mod. Weitergehende Aktionen werden nicht ausgeführt.

8.8.2. Befehle registrieren

Meist möchte man, dass Admin Mod auf einen bestimmten Befehl eine Aktion ausführt. Dafür muss zunächst bei Admin Mod ein Befehl registriert werden. Üblicherweise

macht man dies im Event plugin_init. Zum einfacheren Verständnis erweitern wir das Grundplugin.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public test(HLCommand,HLData,HLUserName,UserIndex) {
   new sData[MAX_DATA_LENGTH];

   convert_string(HLData,sData,MAX_DATA_LENGTH);
   userlist(sData);

   return PLUGIN_HANDLED;
}

public plugin_init() {
   plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
   plugin_registercmd("admin_test","test",ACCESS_ALL,"Userlist anzeigen");

   return PLUGIN_CONTINUE;
}
```

Das Plugin wurde um einen Befehl „plugin_registercmd„ erweitert, der einen Befehl bei Admin Mod registriert.

```
   plugin_registercmd("admin_test","test",ACCESS_ALL,"Userlist anzeigen");
```

Dabei ist admin_test der Befehl, auf den Admin Mod später reagiert und die Funktion „test“ ausführt, die ebenfalls hinzugefügt wurde. Zugriff auf den Befehl haben alle Spieler (ACCESS_ALL). Die vordefinierten Accesslevel sind der admin.inc zu entnehmen oder selbst zu definieren. Zu letzt gibt man einen Hilfetext an, der bei der Ausführung von „admin_help“ ausgegeben wird.

Damit Admin Mod auch eine Aktion ausführen kann, wenn das Event „admin_test“ ausgeführt wird, muss auch die Funktion „test“ definiert werden.

```
public test(HLCommand,HLData,HLUserName,UserIndex) {
   new sData[MAX_DATA_LENGTH];

   convert_string(HLData,sData,MAX_DATA_LENGTH);
   userlist(sData);

   return PLUGIN_HANDLED;
}
```

Damit Admin Mod die Funktion auch aufrufen kann, muss ein „public“ vorangestellt werden. Die übergebenen Variablen bei einem Befehlsevent sind vorgegeben. „HLCommand“ bringt den die Funktion aufrufenden Befehl (hier: „admin_test“). Es können aber

auch verschiedene Befehle die gleiche Funktion aufrufen. Dann kann man mit Verzweigungen unterschiedlichen Code ausführen lassen. In „HLData“ stehen die Befehlsoptionen und in „HLUserName“ der Spielername, der den Befehl aufgerufen hat. „UserIndex“ wiederum gibt an, in welchem Slot der aufrufenden Spieler auf dem Server ist.

Die HL-Variablen müssen, sofern sie genutzt werden sollen, erst in Small-Strings konvertiert werden. Nur der UserIndex liegt bereits als Zahl vor.

```
    convert_string(HLData,sData,MAX_DATA_LENGTH);
```

Nach Ausführung der convert_string Funktion liegt alles, was nach „admin_test“ eingegeben wurde, in der Variable sData vor. Nun wird die Funktion „userlist“ ausgeführt.

```
    userlist(sData);
```

Ist sData leer, werden alle Spieler auf dem Server mit einigen Zusatzinformationen in der Console ausgegeben. Ist z.B. „admin_test a“ eingegeben worden, ist sData = „a“. Es werden dann alle Spieler ausgegeben, deren Name ein „a“ enthält.

Die Funktion wird nicht wie bei der plugin_init durch PLUGIN_CONTINUE beendet.

```
    return PLUGIN_HANDLED;
```

Admin Mod geht der Reihe nach durch die Plugins und überprüft, ob der gesuchte Befehl einem Plugin gehört. Erst wenn es kein passendes Plugin findet, gibt es den Befehl an den Server weiter. Würde man hier PLUGIN_CONTINUE angeben, so würde Admin Mod in den nachfolgenden Plugins weitersuchen und den Befehl anschließend an den Server weitergeben. Der Server kennt den Befehl aber nicht und würde mit „Unknown command“ antworten, obwohl die Admin Mod Aktion durchgeführt wurde.

Mit PLUGIN_HANDLED verhindert man, dass Admin Mod weitersucht bzw. den Befehl an den Server weitergibt.

8.8.3. Auf say reagieren

Die gleiche Funktionalität kann man auch über den Chat erreichen, wenngleich die Ausgabe weiterhin in der Console stattfindet. Es soll also bei der Eingabe von „userlist“ im Chat, die Spielerliste in der Console ausgegeben werden.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public test(HLCommand,HLData,HLUserName,UserIndex) {
   new sData[MAX_DATA_LENGTH];
   new sCommand[MAX_COMMAND_LENGTH];

```

```
    convert_string(HLData,sData,MAX_DATA_LENGTH);
    strsep(sData," ",sCommand,MAX_COMMAND_LENGTH,sData,MAX_DATA_LENGTH);

    if (streq(sCommand, "userlist") == 1) {
       userlist(sData);
       return PLUGIN_HANDLED;
    }

    return PLUGIN_CONTINUE;
}

public plugin_init() {
    plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
    plugin_registercmd("say","test",ACCESS_ALL);
    plugin_registercmd("say_team","test",ACCESS_ALL);
    plugin_registerhelp("say",ACCESS_ALL,"userlist <Muster>: Userlist anzeigen");

    return PLUGIN_CONTINUE;
}
```

Es wurde „admin_test“ durch „say“ (freier Chat) und „say_team“ (Teamchat) ersetzt.

```
    plugin_registercmd("say","test",ACCESS_ALL);
    plugin_registercmd("say_team","test",ACCESS_ALL);
    plugin_registerhelp("say",ACCESS_ALL,"userlist <Muster>: Userlist anzeigen");
```

Da auch andere Plugins „say“ und „say_team“ reagieren können, verzichtet man beim Registrieren auf einen Hilfetext. Stattdessen legt man mit plugin_registerhelp einen Hilfetext fest. Auf diese Weise können auch Unteroptionen genauer beschrieben werden. Eine Funktion muss nicht angegeben werden, da der Befehl nur den Hilfetext festlegt.

Auch die Funktion „test“ hat sich verändert.

```
public test(HLCommand,HLData,HLUserName,UserIndex) {
    new sData[MAX_DATA_LENGTH];
    new sCommand[MAX_COMMAND_LENGTH];

    convert_string(HLData,sData,MAX_DATA_LENGTH);
    strsep(sData," ",sCommand,MAX_COMMAND_LENGTH,sData,MAX_DATA_LENGTH);

    if (streq(sCommand, "userlist") == 1) {
       userlist(sData);
       return PLUGIN_HANDLED;
    }

    return PLUGIN_CONTINUE;
}
```

Da der Befehl immer „say“ oder „say_team“ lautet, muss man erst sData trennen, um an den eigentlichen Befehl heranzukommen.

```
    strsep(sData," ",sCommand,MAX_COMMAND_LENGTH,sData,MAX_DATA_LENGTH);
```

Mit „strsep“ wird der String sData am ersten Leerzeichen in sCommand und sData getrennt (sData wird dabei überschrieben). Man muss jeweils auch die maximale Länge des Strings angeben.

Anschließend wird der Befehl in sCommand überprüft, ob er mit „userlist“ identisch ist (streq). Wenn dies der Fall ist, wird die Spielerliste ausgegeben und nicht weitergesucht (PLUGIN_HANDLED). Das bedeutet auch, dass die Nachricht nicht im Chat ankommt. Admin Mod gibt die Nachricht nicht an den Server weiter. Ist sCommand nicht mit „userlist“ identisch, gibt Admin Mod die Nachricht an das nächste Plugin oder an den Server weiter. Sofern also kein anderes Plugin die Weiterleitung blockiert, landet sie beim Serverchat.

8.8.4. Spielerconnect abfangen

Es könnte von Interesse sein, eine Aktion durchzuführen, wenn ein Spieler den Server betritt. Im folgenden Plugin soll in den Logfiles gespeichert werden, welcher Slot beim Connect besetzt wird.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public plugin_connect(HLUserName, HLIP, UserIndex) {
   new sText[MAX_TEXT_LENGTH];

   snprintf(sText,MAX_TEXT_LENGTH,"Slot %d ist besetzt",UserIndex);
   log(sText);

   return PLUGIN_CONTINUE;
}

public plugin_init() {
   plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
   return PLUGIN_CONTINUE;
}
```

Der Event „plugin_connect“ muss wie alle Events mit „public“ öffentlich gemacht werden.

```
public plugin_connect(HLUserName, HLIP, UserIndex) {
```

Wie beim Serverbefehl-Event gibt es auch für den Spielerbeitritt festgelegte Variablen, die ausgewertet werden können. Leider ist der Event in der Regel zu früh, so dass weder Spielername noch seine IP zurückgegeben werden. Nur der Slot (UserIndex) lässt sich zuverlässig nutzen.

Aus dem „UserIndex“ soll ein ausagekräfter Text für die Logdateien gemacht werden.

```
    snprintf(sText,MAX_TEXT_LENGTH,"Slot %d ist besetzt",UserIndex);
    log(sText);
```

Mit der Funktion „snprintf“ lassen sich einfach Strings und Zahlen in einen bestehenden String implementieren. Neben dem String, in den der Text gespeichert werden soll, muss auch die maximale Länge des Strings angegeben werden. Der folgende String beinhaltet neben dem eigentlichen Text auch einen Platzhalter für den Slot (%d steht für eine Zahl, %s steht für einen String). Abschließend müssen in der Reihenfolge des Auftauchens im Text die Variablen angegeben werden. Das Ergebnis sieht dann z.B. folgendermaßen aus:

```
Slot 9 ist besetzt
```

Als letztes wird der Text mittels der Funktion „log“ in die Logdatei eingetragen. Dies passiert bei jedem Spieler, der auf den Server kommt.

8.8.5. Timer verwenden

Oftmals möchte man bestimmte Aktionen in regelmäßigen Abständen wiederholen lassen. Admin Mod bietet dafür Timer an, die eine Aktion ein- oder mehrmals durchführen können. Es wird damit bei jedem Auslösen des Timers ein Event erzeugt, das mit einer vordefinierten Funktion (wie bei einem Serverbefehl) verarbeitet werden kann.

Das folgende Plugin zeigt in roter, zentrierter Schrift alle 60 Sekunden den Begriff „Admin Mod“ an.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public say_stuff(Timer,Repeat,HLName,HLParam) {
   centersay("Admin Mod",10,255,0,0);
}

public plugin_init() {
   plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
   set_timer("say_stuff", 60, 99999,"");
   return PLUGIN_CONTINUE;
}
```

In plugin_init wird zunächst mittels set_timer ein Timer gestartet.

```
   set_timer("say_stuff", 60, 99999,"");
```

Es muss angegeben werden, welche Funktion („say_stuff“) beim Ablauf des Timers aufgerufen werden soll, wie lang der Timer jeweils laufen soll (60 Sekunden), wie lang er wiederholt werden soll. Die 99999 hat eine Sonderstellung und steht für unendlich

viele Wiederholungen. Admin Mod wird nun versuchen alle 60 Sekunden die Funktion „say_stuff“ auszuführen.

Diese Funktion muss wiederum öffentlich (public) gemacht werden.

```
public say_stuff(Timer,Repeat,HLName,HLParam) {
   centersay("Admin Mod",10,255,0,0);
}
```

Da es sich um eine Funktion handelt, die von einem Timer aufgerufen wurde, gibt es wieder festgelegte Variable. Zunächst wird die Nummer des Timers zurückgegeben. Diese erhält man auch als Rückgabe von set_timer. Weiterhin wird übergeben, um die wievielte Wiederholung es sich handelt. „HLName“ gibt den Spieler zurück, der den Timer ausgelöst hat. Hier ist der Timer automatisch ausgelöst worden, es wird kein Name zurückgegeben. Wird der Timer aus einem Serverbefehlevent ausgeführt, wird der Spieler zurückgegeben, der den Befehl aufgerufen hat. Zuletzt wird ein beliebiger String „HLParam“ übergeben, der in set_timer festgelegt wurde. Im Beispiel wurde nichts angegeben. „HLName“ und „HLParam“ müssen mit convert_string konvertiert werden.

Die centersay Funktion sorgt für die eigentlich Darstellung des Texts „Admin Mod“. Dieser muss natürlich angegeben werden. Weiterhin benötigt die Funktion die Dauer der Darstellung (10 Sekunden) und die Farbe in RGB-Farben (jede Farbe 0 bis 255).

8.8.6. Vote ausführen

Das nächste Beispiel soll zeigen, wie man einen Vote mit Admin Mod durchführt. In diesem Plugin soll die Mehrheit der Spieler entscheiden, ob in einem Counter-Strike Spiel die Map neu gestartet werden soll.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public test(HLCommand,HLData,HLUserName,UserIndex) {
   if (vote_allowed()!=1) {
      selfmessage( "Vote noch nicht erlaubt.");
      return PLUGIN_HANDLED;
   }

   vote("Restart?","Ja","Nein","HandleVsay","");
   return PLUGIN_HANDLED;
}

public HandleVsay(WinningOption,HLData,VoteCount,UserCount) {
   if (WinningOption == 1) {
      say("Mehrheit wollte Restart");
      setstrvar("sv_restart","1");
   }
```

```
21      else {
22         say("Mehrheit wollte keinen Restart");
23      }
24   }
25
26   public plugin_init() {
27      plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
28      plugin_registercmd("admin_test","test",ACCESS_ALL,"Restartvote");
29      return PLUGIN_CONTINUE;
30   }
```

Um den Vote ausführen zu können, braucht man ein Event. Es wurde zu diesem Zweck ein Befehl eingefügt (Zeile 28), der auf die „test“ Funktion verweist. Natürlich lässt sich der Vote auch mit anderen Events koppeln.

Zunächst muss überprüft werden, ob ein Vote überhaupt erlaubt ist.

```
 7      if (vote_allowed()!=1) {
 8         selfmessage( "Vote noch nicht erlaubt.");
 9         return PLUGIN_HANDLED;
10      }
```

Die Funktion vote_allowed überprüft, ob die in der Variablen vote_freq festgelegte Zeit seit dem letzten Vote bzw. des Mapchanges abgelaufen ist. Ist dies nicht der Fall, wird dem Aufrufenden ein Nachricht in die Console gesendet (selfmessage) und anschließend die Ausführung mit „return“ abgebrochen. Diese Routine verhindert, dass der unzulässige Aufruf von vote zu einer Fehlermeldung führt.

Anschließend wird der Vote mit der Funktion vote gestartet. Vote benötigt eine Frage, bis zu neun Optionen, die aufzurufende Funktion und Zusatzdaten.

```
12      vote("Restart?","Ja","Nein","HandleVsay","");
```

Hier wird gefragt, ob ein Restart durchgeführt werden soll, es gibt zwei Optionen (Ja oder Nein), die aufzurufende Funktion heißt „HandleVsay“ und es werden keine Zusatzdaten übertragen. Es ist zu beachten, dass keine Vote-Dauer festgelegt werden kann. Hierzu zieht Admin Mod die Variable amv_vote_duration heran. Auch die Schriftfarbe kann nicht verändert werden (weiß).

Der Vote-Event wird mit Ablauf des Vote-Timers ausgelöst.

```
16   public HandleVsay(WinningOption,HLData,VoteCount,UserCount) {
17      if (WinningOption == 1) {
18         say("Mehrheit wollte Restart");
19         setstrvar("sv_restart","1");
20      }
21      else {
22         say("Mehrheit wollte keinen Restart");
23      }
24   }
```

Einige Variablen werden bei diesem Event mitgeliefert. „WinningOption“ gibt die Nummer der Option an, die gewonnen hat. Dies wird über die Variable admin_vote_ratio vorab festgelegt. „HLData“ gibt die Zusatzoptionen zurück, die hier nicht genutzt wurden (muss konvertiert werden). Außerdem bekommt man die Anzahl der Stimmen, die auf die Option gefallen sind (VoteCount) sowie die Gesamtzahl der Stimmen (UserCount).

Die erste Option „Ja“ ist die „1“. Wenn Option 1 gewonnen hat, gibt Admin Mod im Chat aus (say), dass sich die Mehrheit für einen Restart entschieden hat. Anschließend wird die Variable „sv_restart“ auf 1 gesetzt (setstrvar), was mit einem Restart nach einer Sekunde gleichzusetzen ist. Gewinnt Option 2 wird dies ebenfalls den Spielern mitgeteilt, aber keine weitere Aktion ausgeführt. Ein „return“ ist hier nicht notwendig, da Admin Mod beim Vote-Event keine Rückgabe benötigt.

8.8.7. Vault-File nutzen

Variablen lassen sich bequem in Plugins speichern. Die Inhalte gehen jedoch beim Mapwechsel oder beim Abschalten des Servers verloren. Einige der Standardplugins dürfen sich über Servervariablen glücklich schätzen, die für sie von Admin Mod erstellt wurden. Um auch anderen Programmierern die Möglichkeit zu geben, Einstellungen dauerhaft zu speichern, wurde die vault.ini geschaffen. Mit einigen wenigen Befehlen lassen sich bequem Zahlen und Strings abspeichern. Man sollte diese Funktionen allen Textdateispielereien vorziehen. Sie sind wesentlich schneller als die zum Lesen und Schreiben von Textdateien.

Das folgende Beispiel schreibt einen zufälligen Wert zwischen 0 und 9 in einen Vault-Eintrag. Es wird dabei die Funktion set_vaultnumdata nachgebaut.

```
#include <core>
#include <string>
#include <admin>
#include <adminlib>

public write_entry(HLCommand,HLData,HLUserName,UserIndex) {
   new_set_vaultnumdata("TEST_ENTRY",random(9));
   return PLUGIN_HANDLED;
}

new_set_vaultnumdata(sEntry[],iContent){
   new sContent[MAX_DATA_LENGTH];
   snprintf(sContent,MAX_DATA_LENGTH,"%d",iContent);
   set_vaultdata(sEntry,sContent);
   return PLUGIN_CONTINUE;
}

public plugin_init() {
   plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
   plugin_registercmd("admin_write","write_entry",ACCESS_ALL,"Schreibe Eintrag");
```

```
21      return PLUGIN_CONTINUE;
22   }
```

Auch in diesem Beispiel muss wieder ein Befehl registriert werden (Zeile 20). Um den Quellcode transparenter zu machen, wurde der Nachbau der Funktion set_vaultnumdata in eine interne Funktion ausgelagert.

```
new_set_vaultnumdata(sEntry[],iContent){
   new sContent[MAX_DATA_LENGTH];
   snprintf(sContent,MAX_DATA_LENGTH,"%d",iContent);
   set_vaultdata(sEntry,sContent);
   return PLUGIN_CONTINUE;
}
```

Die neue Funktion erwartet einen String (sEntry) und eine Zahl (iContent) als Eingabe. Die Zahl iContent wird über die Funktion „snprintf“ in einen String umgewandelt. Man kann selbstverständlich auch numtostr verwenden. Anschließend wird mit set_vaultdata der Variablenname (sEntry) und sein Wert (sContent) in die vault.ini geschrieben. Pro forma wurde ein PLUGIN_CONTINUE als Rückgabewert angegeben.

Es ist zu beachten, dass die Funktion nicht öffentlich ist. Sie hat kein „public“ vorangestellt. Dies ist nicht notwendig, da sie ein integraler Bestandteil des Plugins ist, und nur von diesem selbst aufgerufen wird.

Die neue Funktion „new_set_vaultnumdata“ wird aus „write_entry“ aufgerufen.

```
public write_entry(HLCommand,HLData,HLUserName,UserIndex) {
   new_set_vaultnumdata("TEST_ENTRY",random(9));
   return PLUGIN_HANDLED;
}
```

Als Eintrag wurde „TEST_ENTRY“ verwendet und eine weitere Funktion (random) direkt eingebunden, die eine Zahl zwischen 0 und 9 ausgibt. Die direkte Einbindung funktioniert bei allen Funktionen. Da Small aber nur Zahlen als direkten Rückgabewert verwendet, können auf diese Art keine Strings eingebunden werden. Strings die übergeben wurden, sind jedoch automatisch referenziert, d.h. Änderungen in der Child-Funktion werden automatisch an die Parent-Funktion weitergeleitet. Bei Zahlen muss ein kaufmännisches Und voranstellt werden.

```
new_set_vaultnumdata(sEntry[],&iContent){
```

8.8.8. LogD verwenden

In diesem Abschnitt soll beschrieben werden, wie die Admin Mod Events durch weitere ergänzt werden können. Dazu ist das Metamod-Plugin LogD notwendig. Dieses Plugin erkennt Damages in Counter-Strike und schreibt sie in den Chat.

```
#include <core>
```

```
2  #include <string>
3  #include <admin>
4  #include <adminlib>
5
6  #define ACCESS_CONSOLE 131072
7  #define MNL 33
8
9  public damages(HLCommand,HLData,HLUserName,UserIndex) {
10    new sData[MAX_DATA_LENGTH];
11    new sA[MNL];
12    new sV[MNL];
13    new sW[MNL];
14    new sD[MNL];
15
16    convert_string(HLData,sData,MAX_DATA_LENGTH);
17    strsep(sData," ",sA,MNL,sV,MNL,sW,MNL,sD,MNL,sData,MAX_DATA_LENGTH);
18    strsep(sData,"#",sData,MAX_DATA_LENGTH,sD,MNL);
19
20    playerinfo(strtonum(sA),sA,MNL);
21    playerinfo(strtonum(sV),sV,MNL);
22
23    snprintf(sData,MAX_DATA_LENGTH,"%s traf %s mit %s (-%s hp)",sA,sV,sW,sD);
24    say(sData);
25
26    return PLUGIN_HANDLED;
27 }
28
29 public plugin_init() {
30    plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
31    plugin_registercmd("test_damages","damages",ACCESS_CONSOLE);
32
33    exec("logd_reg 58 admin_command test_damages",1);
34    return PLUGIN_CONTINUE;
35 }
```

Das sieht im ersten Moment etwas komplexer aus. Die eigentliche Funktionsweise lässt sich aber schon mittels der plugin_init erklären.

```
29 public plugin_init() {
30    plugin_registerinfo("Testplugin","Ein Testplugin!","1.0");
31    plugin_registercmd("test_damages","damages",ACCESS_CONSOLE);
32
33    exec("logd_reg 58 admin_command test_damages",1);
34    return PLUGIN_CONTINUE;
35 }
```

In Zeile 33 wird ein Serverbefehl abgesetzt. LogD wird mitgeteilt, dass es bei Event 58 (Damages) den Befehl „admin_command test_damages“ ausführen soll. Die „1“ bei der exec sagt lediglich aus, dass die Ausführung der Eventregistrierung mitgelogged wird. Weitere LogD-Events werden im Abschnitt 8.13 vorgestellt.

Damit Admin Mod auch auf den Serverbefehl reagiert, muss der Befehl „test_damages“ bei Admin Mod registriert werden. Der Access Level „ACCESS_CONSOLE“ wurde bewusst über die üblichen Access Level gesetzt (Zeile 6), da kein User diesen Befehl manuell ausführen können darf. Aus dem gleichen Grund wurde auch der Hilfetext weggelassen.

Zusammengefasst wird auf diese Art beim Zufügen von Schaden automatisch die Funktion „damages“ ausgeführt.

Zwei Besonderheiten sind noch in diesem Plugin zu finden.

```
6  #define ACCESS_CONSOLE 131072
7  #define MNL 33
```

Es wird wie beschrieben ein neuer Access Level definiert. Alle Befehle, die von der Serverconsole ausgeführt werden (z.B. über RCon oder anderen Plugins), haben alle theoretischen Rechte. Normalerweise hat kein User den Access Level 131072. Damit wird sichergestellt, dass kein User einen Pseudoevent absetzen kann. Allerdings ist ein solcher Access Level nicht in der admin.inc definiert, so dass man dies selbst durchführen muss.

Der zweite Eintrag ist ein Alias für MAX_NAME_LENGTH, der wegen Darstellungsgründen für dieses Tutorial eingesetzt werden musste. Anderenfalls wäre die Zeile 17 zu lang geraten. Das ist also normalerweise nicht notwendig.

Die Funktion „damages“ nimmt letztlich den von LogD gelieferten String auseinander und setzt ihn neu zusammen.

```
 9  public damages(HLCommand,HLData,HLUserName,UserIndex) {
10     new sData[MAX_DATA_LENGTH];
11     new sA[MNL];
12     new sV[MNL];
13     new sW[MNL];
14     new sD[MNL];
15
16     convert_string(HLData,sData,MAX_DATA_LENGTH);
17     strsep(sData," ",sA,MNL,sV,MNL,sW,MNL,sD,MNL,sData,MAX_DATA_LENGTH);
18     strsep(sD,"#",sData,MAX_DATA_LENGTH,sD,MNL);
19
20     playerinfo(strtonum(sA),sA,MNL);
21     playerinfo(strtonum(sV),sV,MNL);
22
23     snprintf(sData,MAX_DATA_LENGTH,"%s traf %s mit %s (-%s hp)",sA,sV,sW,sD);
24     say(sData);
25
26     return PLUGIN_HANDLED;
27  }
```

Für Damages sieht der String beispielhaft so aus:

`3 1 p228 damage#28 damage_armor#12 health#72 armor#34`.

Der String beginnt mit dem Userindex des Angreifers, gefolgt von dem des Opfers und der genutzten Waffe. Anschließend kommt der Lebensschaden, der Rüstungsschaden, die verbleibenden Lebenspunkte und die verbleibenden Rüstungspunkte des Opfers. Option und Wert sind hier durch Hashes (#) getrennt. Für die Ausgabe werden in diesem Fall nur die ersten vier Werte benötigt.

Zunächst wird der String an den ersten 4 Leerzeichen getrennt (Zeile 17). sData wird hier als Abfalleimer wiederverwendet (der Restsring wird nicht benötigt). Anschließend wird sD am Hash getrennt (Zeile 18). Auch hier ist sData wieder Abfalleimer verwendet. Auf diese Weise spart man sich Speicher, da keine zusätzlichen Variablen definiert werden müssen.

Aus den Userindizes sollen nun Namen ermittelt werden (playerinfo). Dazu müssen die Strings aber zuvor in Zahlen umgewandelt werden strtonum.

Der String wird dann in Zeile 23 neu zusammengesetzt und könnte dann mit obigen LogD String folgendermaßen aussehen:

```
Player(2) traf Player mit p228 (-28 hp)
```

Dieser Text wird abschließend in den Chat geschrieben.

8.8.9. Menüs nutzen

Die Verwendung von Menüs muss unter Admin Mod als höhere Programmierkunst bezeichnet werden. Es gibt diverse Fallstricke, die zu beachten sind. Gerade komplexe Menüstrukturen bedingen ein hohes Abstraktionsvermögen. Auch kann man mit anderen Menüs kollidieren (auch die von Half-Life; z.B. Teamauswahl oder Waffenkauf). Nicht jedes Plugin benötigt ein Menü. Man sollte sich genau überlegen, ob man eines verwenden möchte. Üblicherweise verdoppelt sich der Code. Außerdem muss der Serveradmin die Variable amv_enable_beta um „menu1“ ergänzt haben.

Nachdem genügend Angst gesäht wurde, soll nun ein Beispiel folgen, dass eine simple Verwendung des Menüs demonstrieren soll. Dem Spieler soll nach Eingabe von „admin_test“ in der Console ein Menü präsentiert werden, das ihn auswählen lässt, ob er einen Maprestart nach 10, 5 oder 2 Sekunden durchführen möchte. Als letzte Auswahlmöglichkeit kann er das Menü auch einfach schließen.

```
#include <core>
#include <console>
#include <string>
#include <plugin>
#include <admin>
#include <adminlib>

new giMenu[MAX_PLAYERS];

public test(HLCommand,HLData,HLUsername,UserIndex) {
```

```
   new sUser[MAX_NAME_LENGTH];
   convert_string(HLUsername,sUser,MAX_NAME_LENGTH);
   new sMenu[MAX_DATA_LENGTH]="sv_restart?:^n1. 10s^n2. 5s^n3. 2s^n8. Close";
   giMenu[UserIndex]=1;
   menu(sUser,sMenu,135);
   return PLUGIN_HANDLED;
}

public menuselect(HLCommand,HLData,HLUserName,UserIndex){
   new sOption[MAX_DATA_LENGTH];
   new iOption;
   if(giMenu[UserIndex]){
      convert_string(HLData,sOption,MAX_DATA_LENGTH);
      iOption=strtonum(sOption);
      switch(iOption){
         case 1:{
            setstrvar("sv_restart","10");
         }
         case 2:{
            setstrvar("sv_restart","5");
         }
         case 3:{
            setstrvar("sv_restart","2");
         }
      }
      giMenu[UserIndex]=0;
      return PLUGIN_HANDLED;
   }
   return PLUGIN_CONTINUE;
}

public plugin_disconnect(HLUserName,UserIndex){
   giMenu[UserIndex]=0;
   return PLUGIN_CONTINUE;
}

public plugin_connect(HLUserName,HLIP, UserIndex) {
   giMenu[UserIndex]=0;
   return PLUGIN_CONTINUE;
}

public plugin_init() {
   plugin_registerinfo("Test","Testplugin","1.0");
   plugin_registercmd("admin_test","test",ACCESS_CONFIG,"Ein Menue");
   plugin_registercmd("menuselect","menuselect",ACCESS_ALL);

   return PLUGIN_CONTINUE;
}
```

Zunächst soll wieder mit der plugin_init begonnen werden:

```
public plugin_init() {
```

```
53      plugin_registerinfo("Test","Testplugin","1.0");
54      plugin_registercmd("admin_test","test",ACCESS_CONFIG,"Ein Menue");
55      plugin_registercmd("menuselect","menuselect",ACCESS_ALL);
56
57      return PLUGIN_CONTINUE;
58   }
```

Es werden zwei Befehle registriert. Zum einen ist das der Befehl „admin_test“, der die Menüauswahl öffnet. Damit nicht jeder ihn ausführen kann, wurde er nur demjenigen zugreifbar gemacht, der Accesslevel 512 hat. Zum anderen wird der Befehl „menuselect“ registriert, der für alle Menüauswahlen in Half-Life verwendet wird. Er darf nicht eingeschränkt werden (ACCESS_ALL), da ansonsten selbst das Team nicht auswählbar ist. Beide Registrierungen verweisen auf entsprechende Funktionen (test und menuselect).

```
10   public test(HLCommand,HLData,HLUsername,UserIndex) {
11      new sUser[MAX_NAME_LENGTH];
12      convert_string(HLUsername,sUser,MAX_NAME_LENGTH);
13      new sMenu[MAX_DATA_LENGTH]="sv_restart?:^n1. 10s^n2. 5s^n3. 2s^n8. Close";
14      giMenu[UserIndex]=1;
15      menu(sUser,sMenu,135);
16      return PLUGIN_HANDLED;
17   }
```

Es wird der Username vom HL Format ins Small Format konvertiert. Anschließend wird der Menütext mit einigen Umbrüchen erstellt, damit die Frage und die Optionen jeweils in einer eigenen Zeile stehen. In Zeile 14 wird eine Zelle des globalen Feldes giMenu auf 1 gesetzt. Der Zellenindex wird durch den Userindex definiert. Auf diese Weise kann festgehalten werden, ob sich der jeweilige Spieler gerade in einem Menü befindet. Das Feld giMenu wurde in Zeile 8 definiert und ist in allen Funktionen gültig.

```
8    new giMenu[MAX_PLAYERS];
```

Schlussendlich wird nun beim aufrufenden Spieler ein Menü mit dem angegebenen Text ausgegeben (Zeile 15, menu). Es wird dabei auch festgelegt, welche Menüauswahlen erlaubt sind. Bei den Optionen 1, 2, 3 und 8 ergibt sich:

$$2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{8-1} = 1 + 2 + 4 + 128 = 135$$

Genaueres darüber ist der Funktionsbeschreibung von menu zu entnehmen. Andere Auswahlmöglichkeiten sind geblockt, während das Menü geöffnet ist.

Um die Menüauswahl auszuwerten, wird die Funktion menuselect benötigt.

```
19   public menuselect(HLCommand,HLData,HLUserName,UserIndex){
20      new sOption[MAX_DATA_LENGTH];
21      new iOption;
22      if(giMenu[UserIndex]){
23         convert_string(HLData,sOption,MAX_DATA_LENGTH);
24         iOption=strtonum(sOption);
25         switch(iOption){
26            case 1:{
```

```
27              setstrvar("sv_restart","10");
28           }
29           case 2:{
30              setstrvar("sv_restart","5");
31           }
32           case 3:{
33              setstrvar("sv_restart","2");
34           }
35        }
36        giMenu[UserIndex]=0;
37        return PLUGIN_HANDLED;
38     }
39     return PLUGIN_CONTINUE;
40  }
```

Wichtig zu beachten ist, dass diese Funktion durch alle Auswahlen ausgeführt wird. Daher sollte der Code minimiert werden, der ausgeführt wird, wenn das eigene Plugin nicht gemeint ist. Daher wird zunächst überprüft, ob die die Zelle des Feldes giMenu auf 1 gesetzt ist (Zeile 22), der Spieler sich also im eigenen Menü befindet. Anderenfalls wird der menuselect Befehl durchgelassen. Hier ist auch das PLUGIN_CONTINUE am Ende der Funktion wichtig. Ein PLUGIN_HANDLED würde die weitere Ausführung des menuselect Befehls unterbinden.

Nachdem die Überprüfung positiv verlaufen ist, wird die übergebene Option HLData in sOption konvertiert und anschließend in eine Ganzzahl umgewandelt. Die Switch-Verzweigung verweist auf setstrvar Funktionen mit unterschiedlichen Werten für die Servervariable sv_restart wie sie im Menütext angegeben waren. Wenn diese gesetzt werden, erfolgt ein Restart nach der angegeben Anzahl an Sekunden. Entspricht iOption keiner Option, wird dafür kein Code ausgeführt. Das Menü wird aber nach jedem menuselect automatisch geschlossen. Die Angabe von der Option 8 schließt also nur das Menü.

Dem Feld giMenu muss noch mitgeteilt werden, dass der Spieler das Menü verlassen hat. Daher wird die dem Spieler zugewiesene Zelle auf 0 gesetzt (Zeile 36). Die weitere Ausführung von menuselect wird mit PLUGIN_HANDLED unterbunden, da das Menü abgearbeitet wurde.

Es gibt Fälle, wo der Spieler das Spiel bei offenem Menü beendet. Es verbleibt eine 1 im Feld, die derjenige erbt, der in den gleichen Slot kommt.

```
42  public plugin_disconnect(HLUserName,UserIndex){
43     giMenu[UserIndex]=0;
44     return PLUGIN_CONTINUE;
45  }
46
47  public plugin_connect(HLUserName,HLIP, UserIndex) {
48     giMenu[UserIndex]=0;
49     return PLUGIN_CONTINUE;
50  }
```

Aus diesem Grunde ist es notwendig, plugin_disconnect zu definieren, damit bei jedem Disconnect die Zelle auf 0 gesetzt wird. Zur Sicherheit wird dies auch überlicherweise für plugin_connect wiederholt.

Damit erhält man ein funktionsfähiges Menü. Es lässt sich das Prinzip aber auch beliebig verkomplizieren. Untermenüs, zweite Seiten, Eingaben im Chat als Optionsparameter sind nur einige wenige Beispiele, wie das Menüprinzip schnell große Ausmaße annimmt. Menüs sind aber ein i-Tüpfelchen auf jedem Plugin, das viel Spielerbeteiligung benötigt.

8.9. Includes

Die Includes beinhalten alle von Admin Mod zur Verfügung gestellten Funktion, aber auch solche, die erst aus anderen Funktion gebildet werden (z.B. in adminlib.inc oder math.inc. Die Includes befinden sich im Verzeichnis „scripting/include". Man kann sie wie auch schon den Quellcode der Admin Mod Plugins mit jedem handelsüblichen Editor öffnen.

Um Includes zu nutzen, müssen sie mit der include-Direktive in das Plugin eingebunden werden.

Eine kurze Beschreibung zu den einzelnen Funktionen eines Includes kann manchmal auch als Kommentar dem Quellcode der Includes selber entnommen werden. Die Beschreibung ist jedoch relativ kurz.

Im Weiteren werden die einzelnen Includes kurz vorgestellt und die zugehörigen Funktionen aufgelistet. Die Beschreibung und Beispiele zu einzelnen Funktionen sind der Funktionsreferenz zu entnehmen.

8.9.1. admin.inc

Die admin.inc ist das Basis-Include, welches für alle Plugins benötigt wird. Dies liegt schon allein daran, dass sie das Event plugin_init zur Verfügung stellt. Aber auch andere essentielle Events und Funktionen werden hier definiert.

Weiterhin sind diverse Konstanten an dieser Stelle definiert (z.B. Access Level). Auch augewählte Enums sind dort zu finden.

Es ist daher notwendig das Include in das eigene Plugin einzubinden. Eine Liste der zur Verfügung gestellten Events und Funktionen ist der anschließenden Tabelle zu entnehmen.

access	get_vaultnumdata	plugin_registerinfo
auth	getstrvar	pointto
ban	gettarget	rainbow
censor_words	getteamcount	readfile
centersay	getvar	reload
centersayex	glow	removespawn
changelevel	godmode	resetfile
check_user	help	say
check_words	kick	selfmessage
consgreet	kill_timer	servertime
convert_string	list_maps	set_serverinfo
currentmap	listspawn	set_timer
cvar_exists	log	set_vaultdata
deletefile	look_in_dir	set_vaultnumdata
directmessage	maptime	setstrvar
exec	maxplayercount	slap
execclient	menu	slay
fileexists	message	spawn
filesize	messageex	speakto
get_serverinfo	motd	strtonum
get_timer	movespawn	systemtime
get_userArmor	nextmap	teleport
get_userAuthID	noclip	timeleft
get_userFrags	playercount	typesay
get_userHealth	playerinfo	unban
get_userindex	playsound	userlist
get_userinfo	plugin_command	valid_map
get_userIP	plugin_connect	valid_mapex
get_username	plugin_disconnect	version
get_userorigin	plugin_exec	vote
get_userSessionID	plugin_info	vote_allowed
get_userTeam	plugin_init	writefile
get_userWONID	plugin_registercmd	
get_vaultdata	plugin_registerhelp	

8.9.2. adminlib.inc

Das adminlib Include beinhaltet Funktionen, die nicht von Admin Mod selber zur Verfügung gestellt werden. Vielmehr liegen diese Funktionen als Small-Code vor, der wiederum auf den Basisfunktionen und -operatoren aufbaut. Die Admin Mod Programmierer banden diese Funktionen ein, da sie häufig benötigt wurden jedoch nicht nativ über Admin Mod zur Verfügung standen. Sie sind im Laufe der Zeit auch nicht in Admin Mod übernommen worden.

Man wird dieses Include nicht immer benötigen, aber es ist zu empfehlen das Include stets einzubinden. Funktionen wie numtostr oder max werden sehr häufig verwendet.

check_auth	numtostr	strmatch
min	reject_message	strstripquotes
max	say_command	check_param
execute_command	strbreak	check_immunity
format_command	streq	execclient_all
log_command	strinit	ChangeMap

8.9.3. console.inc

Das Include console ist in Admin Mod mehr oder weniger nutzlos. Die zugehörigen Funktionen interagieren mit Betriebsystemconsole. Es werden Keyboard-Eingaben gelesen und Text ausgegeben. Unter Umständen lassen sich diese Funktionen zu Debuggingzwecken nutzen.

Es ist nicht nötig dieses Include einzubinden.

getchar	getvalue	printf
getstring	print	

8.9.4. core.inc

Das core Include beinhaltet Funktionen, die direkt (also nicht von Admin Mod) von Small zur Verfügung gestellt werden.

Genutzt werden die Funktionen nur selten (z.B. clamp oder random), aber zur Sicherheit sollte man auch dieses Include einbinden.

clamp	heapspace	strpack
deleteproperty	numargs	strunpack
existproperty	random	swapchars
funcidx	setarg	tolower
getarg	setproperty	toupper
getproperty	strlen	

8.9.5. fixed.inc

Das Include fixed stellt die Grundrechenarten für Festkommazahlen und einige Konvertierungsfunktionen zur Verfügung.

Das Include wird nur dann benötigt, wenn man mit Festkommazahlen arbeiten oder Funktionen aus dem math Include nutzen möchte.

fixed:fdiv	fixed:fixed	fixed:fmul
fixed:ffract	fixed:fixedstr	fround

8.9.6. math.inc

Die Nutzung von Festkommazahlen ist eigentlich nur sinnvoll, wenn man auf Funktionen zurückgreifen will, die über die einfachen Grundrechenarten hinausgehen. Mit Ganzzahlen (Integer) ist man auf die Grundrechenarten beschränkt.

Da eine native Implementierung der Funktionen höherer Mathematik in Admin Mod nicht in Aussicht gestellt wurde, entstand dieses Include, welches die Funktionen aus den Grundrechenarten nachbildet. Das Include ist mehr oder weniger ein Proof of Concept, hatte aber auch das Ziel z.B. Statistiken oder Distanzrechnungen in Admin Mod zu ermöglichen. Bis dato ist kein Plugin bekannt, das dieses Include nutzt. Die Genauigkeit der Berechnungen ist ausreichend genau bis mindestens zur zweiten Nachkommastelle (je nach Funktion).

Wenn mit Festkommazahlen gerechnet werden soll, ist dieses Include wahrscheinlich notwendig. Dann muss aber auch das fixed Include eingebunden werden.

distance	fixed:f_cosh	fixed:f_power
fixed:f_abs	fixed:f_cot	fixed:f_powere
fixed:f_arccos	fixed:f_coth	fixed:f_radtodeg
fixed:f_arccot	fixed:f_degtorad	fixed:f_sin
fixed:f_arcosh	fixed:f_euler	fixed:f_sinh
fixed:f_arcoth	fixed:f_faculty	fixed:f_sqrt
fixed:f_arcsin	fixed:f_ln	fixed:f_tan
fixed:f_arctan	fixed:f_log10	fixed:f_tanh
fixed:f_arctan_help	fixed:f_logab	fixed:strtofix
fixed:f_arsinh	fixed:f_max	fixtostr
fixed:f_artanh	fixed:f_min	matherror
fixed:f_cos	fixed:f_pi	

8.9.7. plugin.inc

Das Include plugin hat nur zwei selten verwendete Funktionen.

In der Regel muss dieses Include nicht eingebunden werden.

plugin_checkcommand	plugin_message

8.9.8. string.inc

Das string Include beschäftigt sich mit Stringauswertungen und -manipulationen. Besonders Funktionen wie snprintf machen das Include fast unentbehrlich.

Es ist sinnvoll dieses Include standardmäßig in das eigene Plugin aufzunehmen.

index	strcpy	strrchr
rindex	strcspn	strsep
snprintf	strgsep	strsplit
strcasecmp	strgsplit	strspn
strcasestr	strgtok	strstr
strcasestrx	strgtokrest	strstrx
strcat	strncasecmp	strsubst
strchr	strncat	strtok
strcmp	strncmp	strtokrest
strcount	strncpy	strtrim

8.10. Funktionsreferenz

8.10.1. access

`access( iAccess, sName[]= "" );`

`iAccess`
Typ: Integer (0 - 2147483647)

`sName[]= ""`
Typ: String (33)

Mit der Funktion kann man den Accesslevel der Person abfragen, die entweder direkt die Funktion im Plugin aufgerufen hat (sName braucht dann nicht angegegeben werden) oder der Person, die man mit sName angibt. Man erhält einen Integer mit Wert 1 zurück, wenn die Person den Rechtelevel iAccess besitzt und eine 0, wenn die Person diesen Rechtelevel nicht besitzt.

Beispiel aus plugin_base (Funktion: admin_chat):

```
    for(i=1; i<=maxplayers; i++) {
       strinit(Name);
       if(playerinfo(i,Name,MAX_NAME_LENGTH)==1) {
          if(access(ACCESS_CHAT,Name)!=0) {
             messageex(Name, Text, print_chat);
          }
       }
    }
```

Es werden alle Playerslots abgefragt, ob ein Spieler vorhanden ist. Wenn ja, wird überprüft, ob er den Rechtelevel (ACCESS_CHAT = 64) hat, um Nachrichten, die über admin_chat abgesetzt wurden, lesen zu dürfen. Erst wenn Admin Mod dies auch bestätigt, wird die Nachricht an den Spieler ausgegeben.

Gehört zu: admin.inc

Siehe auch: auth, check_auth

8.10.2. auth

`auth( sName[]= "" );`

`sName[]= ""`
Typ: String (33)

Mit der Funktion auth kann man abfragen, ob eine Person (sName) als Admin authentifiziert wurde (d.h. in der users.ini steht und sich angemeldet hat). Es spielt dabei keine Rolle, welchen Access Level die Person hat. Die Funktion wird selten für eine vereinfachte Administratorüberprüfung genutzt.

Liefert als Integerwert 1 zurück, wenn die Personen authentifiziert wurde. Andernfalls wird eine 0 zurückgeliefert.

Beispiel aus plugin_bk_hltvannounce[2] (Funktion: admin_hltv_connect):

```
      if(auth(User)){
         hltvconnect(Data);
      }
```

Erst wenn der Spieler (User), der den Befehl zum Umleiten eines anderen Spielers (Data) abgesetzt hat, nachweisen kann, dass er in der plugin.ini steht, d.h. authentifiziert ist, wird dieser Spieler auch auf den HLTV umgeleitet.

Gehört zu: admin.inc

Siehe auch: access, check_auth

[2] http://www.adminmod.de/plugins.php?plugin=plugin_bk_hltvannounce

8.10.3. ban

`ban( sPlayer[], iTime, bBanBy = bBanByID );`

`sPlayer[]`
Typ: String (33)

`iTime`
Typ: Integer (0 - 2147483647)

`bBanBy = bBanByID`
Typ: Enum (0=bBanByID; bBanByIP, bBanBoth)

Die Funktion bannt eine Person unter sPlayer angegebene Person. sPlayer kann der Spielername, die SessionID oder die Steam ID sein. iTime ist der Integerwert in Minuten, wie lange der Spieler gebannt werden soll. Der Wert 0 bedeutet, dass die Person permanent vom Server verbannt wird. Wenn als bBanBy die bBanByID angegeben wird, wird die Steam ID gebannt. Bei LAN-Servern, wo es keine eindeutige Steam ID gibt, muss man bBanByIP verwenden. Dies bestimmt auch, in welche Banliste der Eintrag eingefügt wird. Entweder banned.cfg oder listip.cfg. Der Spieler wird zusätzlich sofort vom Server gekickt.

Beispiel aus plugin_base (Funktion: admin_ban):

```
        if(check_immunity(ban_user)==1) {
           snprintf(Text, MAX_TEXT_LENGTH, "You can't ban '%s'.", TargetName);
           messageex(User,Text,print_chat);
        } else {
           ban(ban_user,BanTime,iBanType);
        }
```

Zunächst wird überprüft, ob der zu bannende Spieler evtl. immun gegen Aktionen ihn betreffend ist. Wenn dies der Fall ist, weist eine mehr oder weniger freundliche Meldung den aufrufenden Spieler auf seine nutzlose Aktion hin. Anderenfalls wird der Bann anhand der ID/IP für die angegebene Zeit ausgeführt und der Spieler vom Server geworfen.

Gehört zu: admin.inc

8.10.4. censor_words

`censor_words( sString[] );`

`sString[]`
Typ: String (100)

Die zensierten Wörter werden in der Datei angegeben, die in der adminmod.cfg mittels words_file definiert wird. Man kann mit censor_words einen Text untersuchen, ob sich darin ein zensiertes Wort befindet. Ist eines vorhanden, werden die Buchstaben durch „*“ ersetzt.

Beispiel aus plugin_retribution (Funktion: Handle_say):

```
      if (check_immunity(User)==0) {
         messageex(User, "Swearing not allowed on this server", print_center);

         new SwearMsg[MAX_TEXT_LENGTH];
         censor_words(Data);
      new i;
      new c=strlen(Data);
      for (i=0;i<c;i++) {
        if (Data[i] == '"') {
          Data[i]='^';
        }
      }

      snprintf(SwearMsg, MAX_TEXT_LENGTH, "%s ^"%s^"", Command, Data);
      execclient(User, SwearMsg);
      return PLUGIN_HANDLED;
    }
```

Nach dem Check, dass der schreibende Spieler keine Immunitität besitzt, gibt es einen Hinweis in der Mitte des Bildschirms, dass Fluchen auf dem Server nicht erlaubt ist. Danach wird das Geschriebene zensiert, die Anführungszeichen ausgetauscht und die bereinigte Chat-Nachricht nochmals vom Spieler abgeschickt (execclient). Um zu verhindern, dass die unzensierte Nachricht versandt wird, muss die weitere Abarbeitung der Nachricht mittels PLUGIN_HANDLED unterbunden werden.

Gehört zu: admin.inc

Siehe auch: check_words

8.10.5. centersay

`centersay( sText[], iTime, iRed, iGreen, iBlue );`

`sText[]`
Typ: String (500) (max. Zeilenlänge: 80)

`iTime`
Typ: Integer (0 - 2147483647)

`iRed`
Typ: Integer (0 - 255)

`iGreen`
Typ: Integer (0 - 255)

`iBlue`
Typ: Integer (0 - 255)

Mit dieser Funktion kann man eine bunte Nachricht (sText) für alle Spieler in der Mitte des Bildschirms produzieren. iTime ist die Einblendzeit in Sekunden. iRed ist der Rotanteil, iGreen der Grünanteil und iBlue der Blauanteil in der Nachricht. Die Funktion centersayex ermöglicht die gleiche Funktionalität für einzelne Spieler.

Beispiel aus plugin_base (Funktion: admin_csay):

```
    if (streq(Color,"red")==1) {
      centersay(Message,10,250,10,10);
```

Wenn der Text Color gleich „red“ ist, wird die Nachricht in Message für 10 Sekunden in einen hellem Rot dargestellt. Es ist zu beachten, dass eine Zeile maximal 80 Zeichen lang sein darf. Mit Zeilenumbrüchen sind bis zu 500 Zeichen möglich.

Gehört zu: admin.inc

Siehe auch: centersayex, messageex, rainbow, typesay

8.10.6. centersayex

`centersayex( sUser[], sText[], iTime, iRed, iGreen, iBlue );`

`sUser[]`
Typ: String (33)

`sText[]`
Typ: String (500) (max. Zeilenlänge: 80)

`iTime`
Typ: Integer (0 - 2147483647)

`iRed`
Typ: Integer (0 - 255)

`iGreen`
Typ: Integer (0 - 255)

`iBlue`
Typ: Integer (0 - 255)

Mit dieser Funktion kann man eine bunte Nachricht (sText) für einzelne Spieler in der Mitte des Bildschirms produzieren. iTime ist die Einblendzeit in Sekunden. iRed ist der Rotanteil, iGreen der Grünanteil und iBlue der Blauanteil in der Nachricht. Die Funktion centersay ermöglicht die gleiche Funktionalität für alle Spieler.

Beispiel aus plugin_cw_creator3[3] (Funktion: kickplayers):

```
        if(Team!=TEAM_PROXY){
            get_vaultdata("CWC_PASS",Text,MAX_TEXT_LENGTH);
            snprintf(Text,MAX_TEXT_LENGTH,"password %s",Text);
            execclient(Target,Text);
            centersayex(Target,sMessage,10,68,255,125);
        }
```

Der Auszug stammt aus einer For-Schleife über alle Spieler. Der Server soll für einen Clanwar geschlossen werden. Um Spielern auf einem HLTV nicht das Passwort zu präsentieren, wird das Passwort nur echten Spielern als Centersay gezeigt. Vorab wurde das neue Passwort aus der vault.ini geladen und bei den Spielern gesetzt.

Gehört zu: admin.inc

Siehe auch: centersay, messageex

[3] http://www.adminmod.de/plugins.php?plugin=plugin_cw_creator3

8.10.7. changelevel

`changelevel( sMap[], iIntermissionPause = 0 );`

`sMap[]`
Typ: String (100)

`iIntermissionPause = 0`
Typ: Integer (0 - 2147483647)

Die Funktion lässt den Server zur angegebenen Map (sMap) wechseln. iIntermissionPause ist ein Integerwert in Sekunden, wie lange gewartet werden soll, bis die Map gewechselt wird.

Beispiel aus plugin_base (Funktion: admin_map):

```
    if (valid_map(Data)==1) {
      say_command(User,Command,Data);
      changelevel(Data, 4);
```

Wenn es sich bei Data um eine gültige Map handelt, wird dies öffentlich gemacht und nach 4 Sekunden ein Mapwechsel durchgeführt.

Gehört zu: admin.inc

Siehe auch: ChangeMap

8.10.8. ChangeMap

`ChangeMap( Timer, Repeat, HLUser, HLParam );`

Bei dieser Funktion handelt es sich ein Timer-Event, das früher statt der direkten Verwendung von changelevel genutzt wurde. Inzwischen ist sie nur noch aus Kompatibilitätsgründen vorhanden.

Gehört zu: adminlib.inc

Siehe auch: changelevel

8.10.9. check_auth

`check_auth( iAuthLevel );`

`iAuthLevel`
Typ: Integer (0 - 2147483647)

Die Funktion liefert eine 1 zurück, wenn die Person, die diese Funktion aufgerufen hat, den Accesslevel iAuthlevel besitzt, andernfalls eine 0. Mit der Funktion access können auch die Rechte anderer ermittelt werden.

Beispiel aus plugin_base (Funktion: admin_rcon):

```
if (check_auth(ACCESS_RCON)==0) {
   selfmessage("Laf. Silly bear.");
   return PLUGIN_HANDLED;
}
```

Um zu verhindern, dass jemand auf die Idee kommt aus der Serverconsole den Befehl admin_rcon aufzurufen, was möglich aber unsinnig ist, wurde eine Abfangroutine implementiert. Nur wenn der User den Access Level 65536 (ACCESS_RCON) besitzt, wird der RCon-Befehl über Admin Mod abgesetzt. Anderenfalls wird man etwas unhöflich auf das unsinnige Ansinnen hingewiesen.

Gehört zu: adminlib.inc

Siehe auch: access, auth, check_immunity

8.10.10. check_immunity

`check_immunity( sTarget[] );`

`sTarget[]`
Typ: String (33)

Die Funktion überprüft, ob der Spieler (sTarget) den Rechtelevel für Immunität (4096) besitzt. Dieses und andere Rechtelevel lassen sich auch mit der Funktion access überprüfen.

Beispiel aus plugin_base (Funktion: admin_ban):

```
      if(check_immunity(ban_user)==1) {
         snprintf(Text, MAX_TEXT_LENGTH, "Laf. You can't ban '%s'.", TargetName);
         messageex(User,Text,print_chat);
      } else {
         ban(ban_user,BanTime,iBanType);
      }
```

Zunächst wird überprüft, ob der zu bannende Spieler immun gegen Aktionen ihn betreffend ist. Je nach Ausfallen der Überprüfung wird entweder eine Meldung über den Fehlschlag des Banns oder der Bann selbst abgesetzt.

Gehört zu: adminlib.inc

Siehe auch: access, auth, check_auth

8.10.11. check_param

`check_param( sParam[] );`

`sParam[]`
Typ: String (100)

Die Funktion überprüft, ob der String sParam gleich dem String „on“ ist. Bei „on“ gibt die Funktion 1 anderenfalls 0 zurück.

Beispiel aus plugin_fun (Funktion: admin_fun):

```
if(check_param(Data)==1) {
   execute_command(User,Command,"admin_fun_mode","1");
} else {
   execute_command(User,Command,"admin_fun_mode","0");
   KillGlow();
}
```

Wenn der Befehl admin_fun abgesetzt wurde, wird überprüft, ob als Option „on“ übergeben wurde. Wenn dies der Fall ist, wird der Fun Mode aktiviert, anderenfalls deaktiviert und alle aktiven Glows abgeschaltet.

Gehört zu: adminlib.inc

8.10.12. check_user

`check_user( sPlayer[] );`

`sPlayer[]`
Typ: String (33)

Überprüft, ob sich ein Spieler mit dem Namen (auch Teil), ID oder IP (sPlayer) auf dem Server befindet. Ein positive Überprüfung gibt eine 1 eine negative eine 0 zurück. Die Funktion wird gern für die Bot oder HLTV-Erkennung eingesetzt.

Beispiel aus plugin_base (Funktion: admin_ban):

```
    if (check_user(ban_user)==1) {
      get_username(ban_user,TargetName,MAX_NAME_LENGTH);
```

Es wird überprüft, ob ein entsprechender Spieler auf dem Server existiert. In dem Fall wird über die Funktion get_username aus der ID oder IP ein Name gemacht oder zu einem Namen ergänzt.

Gehört zu: admin.inc

8.10.13. check_words

`check_words( sData[] );`

`sData[]`
Typ: String (100)

Die zensierten Wörter werden in der Datei angegeben, die in der adminmod.cfg mittels words_file definiert wird. Man kann mit check_words einen Text untersuchen, ob sich darin ein zensiertes Wort befindet. Ist eines vorhanden, wird eine 0 zurückgegeben.

Beispiel aus plugin_retribution (Funktion: Handle_say):

```
if(check_words(NewName)==0) {
   execclient(OldName,"name OldName");
   return PLUGIN_HANDLED;
}
```

Wenn der neue Spielername ein verbotenes Wort enthält, wird veranlasst, dass er seinen Namen wieder zurücksetzt.

Gehört zu: admin.inc

Siehe auch: censor_words

8.10.14. clamp

`clamp( value, min=cellmin, max=cellmax );`

`value`
Typ: Integer (-2147483648 - 2147483647)

`min=cellmin`
Typ: Integer (-2147483648 - 2147483647)

`max=cellmax`
Typ: Integer (-2147483648 - 2147483647)

Manchmal ist es gewünscht, dass sich ein Zahl (value) innerhalb eines bestimmten Bereichs befindet. Liegt sie oberhalb des Bereichs wird sie durch die obere Bereichsgrenze (max) bzw. unterhalb durch die untere Bereichsgrenze (min) ersetzt.

Beispiel aus plugin_CS (Funktion: menuselect):

```
                        new Team;
                        get_userTeam(UserName,Team);
                        Team = clamp(Team,1,2) - 1;
```

Zunächst wird das Team des Spielers ermittelt. Anschließend wird zur Vermeidung einer Feldadressierung außerhalb des gültigen Bereichs (4 - AMX_ERR_BOUNDS) die möglichen Teams auf 1 und 2 begrenzt. Zur weiteren Bearbeitung werden die Teamnummern dekrementiert.

Gehört zu: core.inc

Sieh auch: f_max, f_min, max, min

8.10.15. consgreet

`consgreet( sMessage[] );`

`sMessage[]`
Typ: String (256)

Mit der consgreet Funktion kann man Nachrichten (sMessage) in der Konsole des Spielers anzeigen, der sich gerade mit dem Server verbindet (zum Lesen der Serverregeln etc.). Man kann mit der Funktion auch ein Textfile anzeigen lassen. Dann muss sMessage den Pfad und die Textdatei mit Endung txt enthalten.

Inzwischen wird die Console beim Connect nicht mehr angezeigt. Die Funktion ist daher als obsolet anzusehen.

Beispiel aus plugin_dale_consgreet[4] (Funktion: plugin_connect):

```
if(fileexists("consgreet.txt")==1) {
consgreet("================================================================");
consgreet("------------------------Server Stuff--------------------------");
consgreet("consgreet.txt");
consgreet("");
}
```

Zunächst wird überprüft, ob die Datei consgreet.txt existiert. Anschließend wird eine Überschrift generiert und der Inhalt der Datei consgreet.txt in die Console geschrieben. Eine Leerzeile schließt den Code ab.

Gehört zu: admin.inc

Siehe auch: messageex, selfmessage

[4] http://www.adminmod.de/plugins.php?plugin=plugin_dale_consgreet

8.10.16. convert_string

`convert_string( HLString, sSmallString[], iMaxLength );`

`iHLString`
Typ: String (variabel, theoretisch: 2147483647)

`sSmallString[]`
Typ: String (iMaxLength)

`iMaxLength`
Typ: Integer (variabel, theoretisch: 1 - 2147483647)

Admin Mod erhält von der Engine den HLString, der nicht direkt nutzbar ist. Daher muss der Inhalt mit convert_string in einen SmallString umgewandelt werden.

Beispiel aus plugin_base (Funktion: admin_ban):

```
public admin_ban(HLCommand,HLData,HLUserName,UserIndex) {
   new ban_user[MAX_DATA_LENGTH];
   new BanTime = 0;
   new iBanType = bBanByID;
   new Command[MAX_COMMAND_LENGTH];
   new Data[MAX_DATA_LENGTH];
   new strTime[MAX_NUMBER_LENGTH];
   new strType[MAX_NAME_LENGTH];
   new Text[MAX_TEXT_LENGTH];
   new TargetName[MAX_NAME_LENGTH];
   new User[MAX_NAME_LENGTH];

   convert_string(HLCommand,Command,MAX_COMMAND_LENGTH);
   convert_string(HLData,Data,MAX_DATA_LENGTH);
   convert_string(HLUserName,User,MAX_NAME_LENGTH);
```

Nach einigen Variablendeklaration werden die übergebenen HL-Strings (HLCommand, HLData, HLUserName) zur weiteren Bearbeitung in einen Small-String umgewandelt. Dabei wird auf die maximale Stringlänge aus den Deklarationen Rücksicht genommen.

Gehört zu: admin.inc

8.10.17. currentmap

`currentmap( sMap[], iMaxLength );`

`sMap[]`
Typ: String (iMaxLength)

`iMaxLength`
Typ: Integer (variabel, theoretisch: 1 - 2147483647), praktisch: 33)

Mit der Funktion currentmap wird die aktuell laufende Map (sMap) zurückgegeben.

Beispiel aus plugin_chat (Funktion: SayCurrentMap):

```
SayCurrentMap() {
   new Text[MAX_TEXT_LENGTH];
   new CurrentMap[MAX_NAME_LENGTH];

   currentmap(CurrentMap,MAX_NAME_LENGTH);
   snprintf(Text, MAX_TEXT_LENGTH, "The current map is: %s", CurrentMap);
   say(Text);
}
```

Die derzeit auf dem Server gespielte Map wird ermittelt. Die maximale Stringlänge von 33 (MAX_NAME_LENGTH) sollte völlig ausreichend sein. Anschließend wird allen Spielern mitgeteilt, auf welcher Map sie gerade spielen.

Gehört zu: admin.inc

Siehe auch: nextmap

8.10.18. cvar_exists

`cvar_exists( sCvar[] );`

`sMap[]`
Typ: String (variabel, theoretisch: 1 - 2147483647, praktisch: 33)

Die Funktion dient der Überprüfung, ob eine bestimmte Servervariable existiert. U.U. kann man durch diese Funktion herausbekommen, welche HL-Modifikation läuft.

Beispiel aus plugin_CS (Funktion: GetVersion):

```
GetVersion() {
    if( cvar_exists("sv_region") ) {
        return V15 + 1;
    }
    return V15;
}
```

In diesem Fall wird überprüft, ob die Variable „sv_region“ existiert. Diese wurde erst in der Counter-Strike Version 1.6 eingeführt. Da sich einiges im Handling zwischen CS 1.5 und CS 1.6 unterscheidet, müssen einige Funktionen für beide Versionen funktional bleiben. Mittels Verzweigungen auf Basis des Versionschecks kann diese Problematik bequem umschifft werden.

Gehört zu: admin.inc

Siehe auch: nextmap

8.10.19. deletefile

`deletefile( sFilename[] );`

`sFilename[]`
Typ: String (200)

Löscht die Datei (sFilename), wenn sie existiert. Relative Pfade vom Modverzeichnis aus sind erlaubt. Es muss file_access_write auf 1 gesetzt sein, damit die Funktion wirksam wird. Es ist nur möglich Dateien im Modverzeichnis oder darunter zu schreiben und zu löschen.

Beispiel aus plugin_bk_res[5] (Funktion: admin_res_refresh):

```
        if(fileexists(sMap)){
            deletefile(sMap);
        }
```

Zunächst wird überprüft, ob die Datei (sMap) überhaupt existiert. Erst bei einer positiven Überprüfung wird sie gelöscht.

Gehört zu: admin.inc

Siehe auch: fileexists, filesize, readfile, resetfile, writefile

[5] http://www.adminmod.de/plugins.php?plugin=plugin_bk_res

8.10.20. deleteproperty

`deleteproperty( id=0, const name[]="", value=cellmin );`

`id=0`
Typ: Integer (-2147483648 - 2147483647)

`const name[]=""`
Typ: String (variabel)

`value=cellmin`
Typ: Integer (-2147483648 - 2147483647)

Mit dieser Funktion kann eine so genannte Property bei gegebenem Schlüssel gelöscht werden. Es handelt sich dabei um eine Funktion, deren Benutzung unter Admin Mod vermieden werden sollte. Stattdessen wird empfohlen z.B. die Funktionen get_vaultdata oder set_vaultdata zurückzugreifen.

Mehr Informationen zum Thema sind im Abschnitt 8.14 nachzulesen.

Beispiel:

```
deleteproperty(2,"test_prop");
deleteproperty(3,15);
```

Das erste Beispiel löscht die in ID 2 befindliche Property „test_prop“, während das zweite Beispiel die in ID 3 befindliche Property 15 löscht.

Gehört zu: core.inc

Siehe auch: existproperty, getproperty, setproperty

8.10.21. directmessage

```
directmessage( sMessage[], iUserID = -1, uid:tUidType =
uid:uid_SessionID );
```

`sMessage[]`
Typ: String (100)

`iUserID = -1`
Typ: Integer (-1 - 2147483647)

`uid:tUidType = uid:uid_SessionID`
Typ: Enum (0=uid_none, uid_invalid, uid_index, uid_SessionID, uid_wonID)

Die Nachricht sMessage kann direkt an den Spieler mit dem angegebenen Userindex bzw. der Session ID geschickt werden. Ob es sich um den Userindex oder um die Session ID handelt, muss mit „uid_index“ bzw. „uid_SessionID“ angegeben werden. Die weiteren Optionen „uid_none“, „uid_invalid“, „uid_wonID“ werden nicht (mehr) genutzt.

Beispiel aus plugin_base (Funktion: admin_dmsg):

```
179     switch( sType[0] ) {
180     case 'i':
181        tType = uid_index;
182     case 's':
183        tType = uid_SessionID;
184     case 'w':
185        tType = uid_wonID;
186     default:
187        tType = uid_invalid;
188     }  // switch()
189
190
191     strstripquotes(sMessage);
192     directmessage( sMessage, iUid, tType );
```

Abhängig vom Buchstaben, der sich in der ersten Zelle von sType befindet, wird der UID Typ festgelegt. Die um Anführungszeichen befreite Nachricht (Zeile 191: strstripquotes) wird der ID (Session ID oder Userindex) geschickt.

Gehört zu: admin.inc

Siehe auch: centersayex, message, messageex, selfmessage

8.10.22. distance

`distance( x1, x2, y1=0, y2=0, z1=0, z2=0 );`

`x1`
Typ: Integer (-2147483648 - 2147483647)

`x2`
Typ: Integer (-2147483648 - 2147483647)

`y1=0`
Typ: Integer (-2147483648 - 2147483647)

`y2=0`
Typ: Integer (-2147483648 - 2147483647)

`z1=0`
Typ: Integer (-2147483648 - 2147483647)

`z2=0`
Typ: Integer (-2147483648 - 2147483647)

Die Funktion „distance“ berechnet die Entfernung zwischen zwei Punkten auf der Map.

Bei großen Entfernungen zwischen zwei Punkten, kann es zu Problemen kommen. Die Berechnung ermittelt die Summe der Quadrate der Abstände in x-, y- und z-Richtung. Sollte diese Summe größer als 2147483647 sein, kommt es zu einem Overflow, der zwar das Plugin nicht abstürzen lässt, aber falsche Ergebnisse liefert. Es wird daher empfohlen die Summe vorab abzuschätzen (z.B. sollte die Summe der Abstände 46340 nicht überschreiten). Man kann dann die Punkte durch einen festen Faktor teilen und ihn nach der Berechnung wieder aufschlagen.

Beispiel aus plugin_sdal_logd_hp50[6] (Funktion: get_user_distance):

```
get_userorigin(Attacker,ax,ay,az);
get_userorigin(Victim,vx,vy,vz);
g_PlayerEnemyDistance[iVID]= distance(ax,vx,ay,vy,az,vz);
```

Der Funktion „get_user_distance“ wird der Name des Siegers und des Verlierers des Duells und der Userindex des Verlierers übergeben. Mittels get_userorigin werden die Positionen der Spieler auf der Map ermittelt und der Abstand zwischen den Spielern in einer Zelle eines Feldes gespeichert.

Gehört zu: math.inc

Siehe auch: get_userorigin

[6] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.23. exec

`exec( sCommand[], bWriteLogEntry = 1 );`

`sCommand[]`
Typ: String (256)

`bWriteLogEntry = 1`
Typ: Integer (0 - 1)

Führt einen beliebigen Serverbefehl aus, kann aber auch Variablen setzen. Ausgenommen sind Admin Mod Befehle. Diese müssen mit plugin_exec ausgeführt werden, damit das Rechtesystem nicht über Plugins ausgehebelt werden kann.

Mit bWriteLogEntry = 0 kann man bewirken, dass keine Logzeile, beginnend mit „[ADMIN]Executing command:“ in die Logs eingetragen wird.

Beispiel aus plugin_base (Funktion: admin_pass):

```
    snprintf(Msg, MAX_DATA_LENGTH, "sv_password %s", Data);
    exec(Msg);
```

Der Inhalt aus „Data“ wird „sv_password“ angehängt und ausgeführt, so dass nun das Passwort gilt, das in „Data“ steht. Auf die Logentry Einstellung wurde verzichtet, so dass die Ausführung geloggt wird.

Gehört zu: admin.inc

Siehe auch: execute_command, plugin_exec, setstrvar

8.10.24. execclient

`execclient( sPlayer[], sCommand[] );`

`sPlayer[]`
Typ: String (100)

`sCommand[]`
Typ: String (100)

Führt einen Befehl beim Spieler aus. Dabei muss der Spielername und der auszuführende Befehl angegeben werden. In der adminmod.cfg muss allow_client_exec auf 1 gesetzt sein.

Beispiel aus plugin_retribution (Funktion: admin_bury):

```
        snprintf(Text, MAX_TEXT_LENGTH, "%s has broke the rules!", TargetName);
        say(Text);

        messageex(TargetName, "Please obey our rules!", print_chat);
        execclient(TargetName, "say Help! I'm stuck!");
```

Allen Spielern wird im Chat mitgeteilt, dass der eingegrabene Spieler gegen die Regeln verstoßen hat. Dieses wird ihm ebenfalls im Chat gesagt. Zusätzlich lässt man den Spieler mittels execclient „Help! I'm stuck!“ sagen.

Gehört zu: admin.inc

Siehe auch: execclient_all

8.10.25. execclient_all

`execclient_all( sCommand[] );`

`sCommand[]`
Typ: String (100)

Führt bei allen Spielern den selben Befehl aus. allow_client_exec 1 muss dazu in der adminmod.cfg gesetzt sein.

Beispiel aus plugin_cw_creator/plugin_logd_cwcaddon[7] (Funktion: logd_cwc_score):

```
        if(maxrounds==0){
           execclient_all("timeleft");
        }
```

Falls die Variable „maxrounds“ auf 0 gesetzt ist, wird bei allen Spielern der Befehl „timeleft“ ausgeführt, der die verbleibende Restzeit auf der Map beim Spieler anzeigt.

Gehört zu: adminlib.inc

Siehe auch: execclient

[7] http://www.adminmod.de/plugins.php?plugin=plugin_cw_creator

8.10.26. execute_command

`execute_command( sUser[], sCommand[], sHalfLifeCmd[], sData[] );`

`sUser[]`
Typ: String (33)

`sCommand[]`
Typ: String (30)

`sHalfLifeCmd[]`
Typ: String (30)

`sData[]`
Typ: String (200)

Führt einen Befehl auf dem Server aus und gibt zusätzlich eine formatierte Nachricht am Bildschirm oder in den Logdateien aus, wenn admin_quiet entsprechend in der adminmod.cfg gesetzt ist.

Zum Ausführen wird der Adminname (sUser), der Admin Mod Befehl (sCommand), den er ausgeführt hat, der zu setzende Serverbefehl bzw. die Servervariable (sHalfLifeCmd) und die Option (sData) benötigt.

Beispiel aus plugin_base (Funktion: admin_hostname):

```
243     convert_string(HLCommand,Command,MAX_COMMAND_LENGTH);
244     convert_string(HLData,Data,MAX_DATA_LENGTH);
245     convert_string(HLUserName,User,MAX_NAME_LENGTH);
246     snprintf(sHostName, MAX_DATA_LENGTH, "^"%s^"", Data);
247     execute_command(User,Command,"hostname",sHostName);
```

Zunächst werden die Half-Life Strings konvertiert. „Data“ sollte dabei den neuen Servernamen beinhalten und wird wegen möglicher Leerzeichen in Anführungszeichen gesetzt (Zeile: 246). Anschließend werden die Daten „execute_command“ übergeben. „User“ und „Command“ geben ausführlich Auskunft darüber, wer welchen Befehl ausgeführt hat.

Gehört zu: adminlib.inc

Siehe auch: exec, setstrvar

8.10.27. existproperty

`existproperty( id=0, const name[]="", value=cellmin );`

`id=0`
Typ: Integer (-2147483648 - 2147483647)

`const name[]=""`
Typ: String (variabel)

`value=cellmin`
Typ: Integer (-2147483648 - 2147483647)

Mit dieser Funktion kann die Existenz einer so genannten Property bei gegebenem Schlüssel überprüft werden. Es handelt sich dabei um eine Funktion, deren Benutzung unter Admin Mod vermieden werden sollte. Stattdessen wird empfohlen z.B. die Funktionen get_vaultdata oder set_vaultdata zurückzugreifen.

Mehr Informationen zum Thema sind im Abschnitt 8.14 nachzulesen.

Beispiel:

```
bBoolean = existproperty(2,"test_prop");
bBoolean = existproperty(3,15);
```

Das erste Beispiel überprüft, ob die in ID 2 befindliche Property „test_prop" existiert, während das zweite Beispiel überprüft, ob die in ID 3 befindliche Property 15 gesetzt wurde.

Gehört zu: core.inc

Siehe auch: deleteproperty, getproperty, setproperty

8.10.28. fileexists

`fileexists( sFilename[] );`

`sFilename[]`
Typ: String (100)

Überprüft unter Angabe von „sFilename“, ob die Datei vorhanden ist. Dazu muss die Variable file_access_read auf 1 in der adminmod.cfg gesetzt sein. Als Rückgabewert erhält man eine 1, wenn die Datei vorhanden, anderenfalls bekommt man eine 0 zurück.

Beispiel aus plugin_base (Funktion: plugin_init):

```
	currentmap(strMap, MAX_DATA_LENGTH);
	snprintf(ExecCommand, MAX_DATA_LENGTH, "%s.cfg", strMap);
	if ( fileexists(ExecCommand) ) {
	   snprintf(ExecCommand, MAX_DATA_LENGTH, "exec %s.cfg", strMap);
	   log(ExecCommand);
	   exec(ExecCommand);
	}
```

Es wird die aktuelle Map ermittelt und überprüft, ob sich im Modverzeichnis eine Konfiguartionsdatei für diese spezielle Map befindet. Existiert die Datei, wird sie ausgeführt. Dies ist eine oftmals übersehene Funktion Admin Mods, die der Basisfunktionalität des HL-Servers zugeschlagen wird.

Gehört zu: admin.inc

Siehe auch: deletefile, filesize, readfile, resetfile, writefile

8.10.29. filesize

`filesize( sFilename[], fsize_unit:Unit = 1 );`

`sFilename[]`
Typ: String (100)

`fsize_unit:Unit = 1`
Typ: Enum (0=bytes, lines)

Man erhält die Anzahl der Zeilen oder die Größe in Byte der unter sFilename angegebenen Datei. Wird „fsize_unit“ nicht angegeben, wird die Anzahl der Zeilen bei „bytes“ hingegen die Dateigröße in Bytes zurückgegeben.

Zum Ausführen der Funktion muss file_access_read auf 1 in der adminmod.cfg gesetzt sein.

Beispiel aus plugin_bk_cron[8] (Funktion: admin_cron_refresh):

```
if(fileexists(filename)==0){
   /* log("plugin_bk_cron found no schedule.ini."); */
   log(cron_nosched);
   return PLUGIN_CONTINUE;
}
sizeoffile=filesize(filename,lines);
```

Ist die unter „filename“ angegebene Datei nicht existent, wird eine Fehlermeldung in den Logdateien ausgegeben und die weitere Ausführung des Befehls abgebrochen. Wird die Datei hingegen gefunden, wird die Anzahl der Zeilen ausgelesen und in die Variable „sizeoffile“ geschrieben.

Gehört zu: admin.inc

Siehe auch: deletefile, fileexists, readfile, resetfile, writefile

[8] http://www.adminmod.de/plugins.php?plugin=plugin_bk_cron

8.10.30. fixed:f_abs

`f_abs( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_abs gibt den Absolutwert einer Festkommazahl aus.

Beispiel aus math.inc (Funktion: f_sqrt):

```
    if(f_abs(fNum)!=fNum){
       iError=1;
       return -1.000;
    }
```

Hier wird die Funktion f_abs benutzt, um zu überprüfen, ob der Wert fNum negativ ist. Ist er negativ, wird die Funktion abgebrochen und -1 zurückgegeben, um zu verhindern, dass die Wurzel aus einer negativen Zahl gezogen wird.

Gehört zu: math.inc

8.10.31. fixed:f_arccos

`f_arccos( fixed:fNum, &iError=0 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arccos gibt den Arkuskosinus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:

```
fNum = f_arccos(fNum,iError);
matherror(iError);
```

Aus fNum wird der Arkuskosinus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_arccot, f_arcsin, f_arctan

8.10.32. fixed:f_arccot

```
f_arccot( fixed:fNum );
```

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_arccot gibt den Arkuskotangens Wert als eine Festkommazahl zurück.

Beispiel:
```
fNum = f_arccot(fNum);
```

Aus fNum wird der Arkuskotangens gebildet.

Gehört zu: math.inc

Siehe auch: f_arccos, f_arcsin, f_arctan

8.10.33. fixed:f_arcosh

```
f_arcosh( fixed:fNum, &iError=0 );
```

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arcosh gibt den Areakosinus Hyperbolicus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:
```
fNum = f_arcosh(fNum,iError);
matherror(iError);
```

Aus fNum wird der Areakosinus Hyperbolicus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_arcoth, f_arsinh, f_artanh

8.10.34. fixed:f_arcoth

`f_arcoth( fixed:fNum, &iError=0 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arcoth gibt den Areatangens Hyperbolicus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:

```
fNum = f_arcoth(fNum,iError);
matherror(iError);
```

Aus fNum wird der Areatangens Hyperbolicus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_arcosh, f_arsinh, f_artanh

8.10.35. fixed:f_arcsin

`f_arcsin( fixed:fNum, &iError=0 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arcsin gibt den Arkussinus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel aus math.inc (Funktion: f_arccos):

```
    return fdiv(f_pi(),2.000)-f_arcsin(fNum);
```

Der Arkuskosinus wird aus der Berechnung $\arccos x = \frac{\pi}{2} - \arcsin x$ gebildet.

Gehört zu: math.inc

Siehe auch: f_arccos, f_arccot, f_arctan

8.10.36. fixed:f_arctan

`f_arctan( fixed:fNum, &iError=0 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arctan gibt den Arkustangens Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel aus math.inc (Funktion: f_arccot):

```
stock fixed:f_arccot(fixed:fNum){
   return fdiv(f_pi(),2.000)-f_arctan(fNum);
}
```

Der Arkuskotangens wird aus der Berechnung $\operatorname{arccot} x = \frac{\pi}{2} - \arctan x$ gebildet.

Gehört zu: math.inc

Siehe auch: f_arccos, f_arccot, f_arcsin

8.10.37. fixed:f_arctan_help

`f_arctan_help( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_arctan_help ist lediglich eine Unterfunktion von f_arctan und ist für eine Verwendung in einem Plugin nicht sinnvoll.

Gehört zu: math.inc

Siehe auch: f_arctan

8.10.38. fixed:f_arsinh

```
f_arsinh( fixed:fNum, &iError=0 );
```

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_arsinh gibt den Areasinus Hyperbolicus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:

```
fNum = f_arsinh(fNum,iError);
matherror(iError);
```

Aus fNum wird der Areasinus Hyperbolicus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_arcosh, f_arcoth, f_artanh

8.10.39. fixed:f_artanh

```
f_artanh( fixed:fNum, &iError=0 );
```

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_artanh gibt den Areatangens Hyperbolicus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:

```
fNum = f_artanh(fNum,iError);
matherror(iError);
```

Aus fNum wird der Areatangens Hyperbolicus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_arcosh, f_arcoth, f_arsinh

8.10.40. fixed:f_cos

`f_cos( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_cos gibt den Kosinus Wert als eine Festkommazahl zurück.

Beispiel aus math.inc (Funktion: f_tan):

```
stock fixed:f_tan(fixed:fNum){
   return fdiv(f_sin(fNum),f_cos(fNum));
}
```

Der Tangens wird aus der Berechnung $\tan x = \frac{\sin x}{\cos x}$ gebildet.

Gehört zu: math.inc

Siehe auch: f_cot, f_sin, f_tan

8.10.41. fixed:f_cosh

`f_cosh( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_cosh gibt den Kosinus Hyperbolicus Wert als eine Festkommazahl zurück.

Beispiel aus math.inc (Funktion: f_tanh):

```
stock fixed:f_tanh(fixed:fNum){
   return fdiv(f_sinh(fNum),f_cosh(fNum));
}
```

Der Tangens Hyperbolicus wird aus der Berechnung $\tanh x = \frac{\sinh x}{\cosh x}$ gebildet.

Gehört zu: math.inc

Siehe auch: f_coth, f_sinh, f_tanh

8.10.42. fixed:f_cot

`f_cot( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_cot gibt den Kotangens Wert als eine Festkommazahl zurück.

Beispiel:
```
fNum = f_cot(fNum);
```

Aus fNum wird der Kotangens gebildet.

Gehört zu: math.inc

Siehe auch: f_cos, f_sin, f_tan

8.10.43. fixed:f_coth

`f_coth( fixed:fNum, &iError=0 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_coth gibt den Kotangens Hyperbolicus Wert als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:
```
fNum = f_coth(fNum,iError);
matherror(iError);
```

Aus fNum wird der Kotangens Hyperbolicus gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_cosh, f_sinh, f_tanh

8.10.44. fixed:f_degtorad

```
f_degtorad( fixed:fNum );
```

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_degtorad rechnet einen Winkel von Grad in Radiant um.

Beispiel:

```
fNum = f_degtorad(fNum);
```

Der Winkel fNum wird in Radiant umgerechnet.

Gehört zu: math.inc

Siehe auch: f_radtodeg

8.10.45. fixed:f_euler

```
f_euler( );
```

Die Funktion f_euler gibt die Eulersche Zahl (2,718) zurück.

Beispiel:

```
fEuler = f_euler();
```

In die Variable fEuler wird die Eulersche Zahl geschrieben.

Gehört zu: math.inc

Siehe auch: f_pi

8.10.46. fixed:f_faculty

`f_faculty( fixed:fValue, &iError=0 );`

`fixed:fValue`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_faculty gibt die Fakultät einer Zahl als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel:

```
fNum = f_faculty(fNum,iError);
matherror(iError);
```

Aus fNum wird die Fakultät ermittelt und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

8.10.47. fixed:f_ln

`f_ln( fixed:fValue, &iError=0 );`

`fixed:fValue`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_ln ermittelt den natürlichen Logarithmus einer Zahl und gibt diesen als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel aus math.inc (Funktion: f_arsinh):

```
    return f_ln(fNum+f_sqrt(fmul(fNum,fNum)+1.000));
```

Der Areasinus Hyperbolicus wird aus der Berechnung $\operatorname{arsinh} x = \ln(x + \sqrt{x^2 + 1.000})$ gebildet.

Gehört zu: math.inc

Siehe auch: f_log10, f_logab

8.10.48. fixed:f_log10

`f_log10( fixed:fValue, &iError=0 );`

`fixed:fValue`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_log10 ermittelt den Zehner-Logarithmus einer Zahl und gibt diesen als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben.

Beispiel aus math.inc (Funktion: f_logab):

```
    fBase=f_log10(fBase,iError);
    if(iError>0){
       return fBase;
    }
```

Die Basis für den beliebigen Logarithmus wird mit dem Zehnerlogarithmus vorbereitet. Ist ein Fehler aufgetreten, wird die Ausführung abgebrochen.

Gehört zu: math.inc

Siehe auch: f_ln, f_logab

8.10.49. fixed:f_logab

```
f_log10( fixed:fBase, fixed:fValue, &iError=0 );
```

`fixed:fBase`
Typ: Fixed (-2147482 - 2147482)

`fixed:fValue`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_logab ermittelt den Logarithmus einer Zahl auf einer beliebigen Basis und gibt diesen als eine Festkommazahl zurück. Darüber hinaus wird bei einer fehlgeschlagenen Berechnung ein Fehlerwert iError zurückgegeben. Benötigt man den natürlichen oder den Zehnerlogarithmus, sollte man auf die Funktionen f_ln und f_log10 zurückgreifen.

Beispiel:

```
fNum = f_logab(fBasis,fNum,iError);
matherror(iError);
```

Aus fNum wird der Logarithmus auf der Basis fBasis gebildet und anschließend eine mögliche Fehlermeldung mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_ln, f_log10

8.10.50. fixed:f_max

`f_max( fixed:fNum, fixed:fNum2 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`fixed:fNum2`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_max gibt aus zwei Festkommazahlen die größte aus. Für Ganzzahlen kann man max nutzen.

Beispiel aus math.inc (Funktion: f_arctan):

```
    fRange=f_max(fNum,0.000);
```

Es wird sichergestellt, dass keine negativen Werte auftreten können. Negative Werte werden auf 0 gesetzt.

Gehört zu: math.inc

Siehe auch: clamp, f_min, max, min

8.10.51. fixed:f_min

`f_min( fixed:fNum, fixed:fNum2 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`fixed:fNum2`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_min gibt aus zwei Festkommazahlen die kleinste aus. Für Ganzzahlen kann man min nutzen.

Beispiel:
```
fMin = f_min(fNum1,fNum2);
```

Der kleinste Wert aus fNum1 und fNum2 wird an fMin übergeben.

Gehört zu: math.inc

Siehe auch: clamp, f_max, max, min

8.10.52. fixed:f_pi

`f_pi( );`

Die Funktion f_pi gibt die Zahl Pi (3,142) zurück.

Beispiel:
```
fPi = f_pi();
```
In die Variable fPi wird die Zahl Pi geschrieben.

Gehört zu: math.inc

Siehe auch: f_euler

8.10.53. fixed:f_power

`f_power( fixed:fBasis, fixed:fExponent, &iError=0 );`

`fixed:fBasis`
Typ: Fixed (-2147482 - 2147482)

`fixed:fExponent`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_power exponiert (fExponent) den in fBasis gegebenen Wert. Will man nur die Eulerzahl exponieren, sollte man auf f_powere zurückgreifen.

Beispiel:
```
fNum = f_power(fBasis,fExponent,iError);
matherror(iError);
```
Die Basis (fBasis) wird mit fExponent exponiert und das Ergebnis in fNum geschrieben. Eine mögliche Fehlermeldung wird mit matherror in die Logdateien geschrieben.

Gehört zu: math.inc

Siehe auch: f_powere

8.10.54. fixed:f_powere

`f_powere( fixed:fExponent, &iError=0 );`

`fixed:fExponent`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

Die Funktion f_power exponiert (fExponent) den die Eulersche Zahl. Will man eine beliebige Zahl exponieren, sollte man auf f_power zurückgreifen.

Beispiel aus math.inc (Funktion: f_sinh):

```
stock fixed:f_sinh(fixed:fNum){
   return fdiv(f_powere(fNum)-f_powere(fmul(fNum,-1.000)),2.000);
}
```

Der Sinus Hyperbolicus wird aus der Berechnung $\sinh x = \frac{e^x - e^{-x}}{2}$ gebildet.

Gehört zu: math.inc

Siehe auch: f_power

8.10.55. fixed:f_radtodeg

`f_radtodeg( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_radtodeg rechnet einen Winkel von Radiant in Grad um.

Beispiel:

```
fNum = f_radtodeg(fNum);
```

Der Winkel fNum wird in Grad umgerechnet.

Gehört zu: math.inc

Siehe auch: f_degtorad

8.10.56. fixed:f_sin

`f_sin( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_sin gibt den Sinus Wert als eine Festkommazahl zurück.

Beispiel aus math.inc (Funktion: f_cos):

```
stock fixed:f_cos(fixed:fNum){
   return f_sin(fNum+fdiv(f_pi(),2.000));
}
```

Der Kosinus wird aus der Berechnung $\cos x = \sin(x + \frac{\pi}{2})$ gebildet.

Gehört zu: math.inc

Siehe auch: f_cos, f_cot, f_tan

8.10.57. fixed:f_sinh

`f_sinh( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_sinh gibt den Sinus Hyperbolicus Wert als eine Festkommazahl zurück.

Beispiel aus math.inc (Funktion: f_tanh):

```
stock fixed:f_tanh(fixed:fNum){
   return fdiv(f_sinh(fNum),f_cosh(fNum));
}
```

Der Tangens Hyperbolicus wird aus der Berechnung $\tanh x = \frac{\sinh x}{\cosh x}$ gebildet.

Gehört zu: math.inc

Siehe auch: f_cosh, f_coth, f_tanh

8.10.58. fixed:f_sqrt

`f_sqrt( fixed:fNum, &iError=0, fixed:fNumStart=1.000 );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

`iError=0`
Typ: Integer (0 - 3)

`fixed:fNumStart`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_sqrt zieht aus einer gegebenen Zahl fNum die Quadratwurzel. Da die Ermittlung der Quadratwurzel eine iterative Berechnung ist, kann die Berechnung beschleunigt werden, in dem ein sinnvoller Startwert für fNumStart angegeben wird (Standard: 1.000). Außerdem wird über iError ein Fehlerwert zurückgegeben.

Beispiel aus math.inc (Funktion: f_arsinh):

```
    return f_ln(fNum+f_sqrt(fmul(fNum,fNum)+1.000));
```

Der Areasinus Hyperbolicus wird aus der Berechnung $\operatorname{arsinh} x = \ln(x + \sqrt{x^2 + 1.000})$ gebildet.

Gehört zu: math.inc

8.10.59. fixed:f_tan

`f_tan( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_tan gibt den Tangens Wert als eine Festkommazahl zurück.

Beispiel:
```
fNum = f_tan(fNum);
```

Aus fNum wird der Tangens gebildet.

Gehört zu: math.inc

Siehe auch: f_cos, f_cot, f_sin

8.10.60. fixed:f_tanh

`f_tanh( fixed:fNum );`

`fixed:fNum`
Typ: Fixed (-2147482 - 2147482)

Die Funktion f_tanh gibt den Tangens Hyperbolicus Wert als eine Festkommazahl zurück.

Beispiel:
```
fNum = f_tanh(fNum);
```

Aus fNum wird der Tangens Hyperbolicus gebildet.

Gehört zu: math.inc

Siehe auch: f_cosh, f_coth, f_sinh

8.10.61. fixed:fdiv

`fixed:fdiv( fixed:dividend, fixed:divisor );`

`fixed:dividend`
Typ: Fixed (-2147482 - 2147482)

`fixed:divisor`
Typ: Fixed (-2147482 - 2147482)

Die Funktion fdiv führt eine Divison zweier Zahlen durch. Theoretisch steht auch „/" als Operatorzeichen zur Verfügung. Die Erkennung durch den Compiler ist leider nicht zuverlässig, so dass die Funktion fdiv empfohlen wird.

Beispiel aus math.inc (Funktion: f_cos):

```
stock fixed:f_cos(fixed:fNum){
   return f_sin(fNum+fdiv(f_pi(),2.000));
}
```

Hier wird der Kosinus durch die Berechnung $\cos x = \sin(x + \frac{\pi}{2})$ ermittelt.

Gehört zu: fixed.inc

Siehe auch: fmul

8.10.62. fixed:ffract

`fixed:ffract( fixed:value );`

`fixed:value`
Typ: Fixed (-2147482 - 2147482)

Mit ffract werden die drei Nachkommastellen als Zahl ermittelt. Dabei wird auch das Vorzeichen des Gesamtwertes übertragen.

Beispiel aus math.inc (Funktion: fixtorstr):

```
iFrac=ffract(fNumber);
if(iFrac<0){
   iFrac*=-1;
   iSign=-1;
}
```

Im Beispiel werden die Nachkommastellen als Zahl ermittelt. Ist die Zahl negativ wird sie negiert und das Vorzeichen in iSign gespeichert.

Gehört zu: fixed.inc

8.10.63. fixed:fixed

`fixed:fixed( value );`

`value`
Typ: Integer (-2147482 - 2147482)

Die Funktion fixed wandelt eine Ganzzahl in eine Festkommazahl um.

Beispiel aus math.inc (Funktion: f_faculty):

```
iValue=fround(fValue);
fValue=1.000;
for(new i=2;i<=iValue;i++){
   fValue=fmul(fValue,fixed(i));
}
```

Zunächst wird zur Ermittlung der Fakultät der Wert auf eine Ganzzahl gerundet. Der Ausgangswert wird auf 1 gesetzt. Die Fakultät wird durch eine For Schleife durch Multiplikation ermittelt. Dabei muss das aktuelle Inkrement bei jedem Durchlauf in eine Festkommazahl mittels der Funktion fixed umgewandelt werden.

Gehört zu: fixed.inc

8.10.64. fixed:fixedstr

`fixed:fixedstr( const string[] );`

`string`
Typ: String (20)

Die Funktion fixedstr wandelt einen String in eine Festkommazahl um. Dabei geht aber ein mögliches Vorzeichen verloren. Das Ergebnis ist immer positiv. Daher wird empfohlen statt fixedstr die Funktion strtofix zu verwenden, die auch das Vorzeichen berücksichtigt.

Beispiel aus math.inc (Funktion: strtofix):

```
if(strtrim(sNumber,"-",0)>0){
   fNumber=fmul(fixedstr(sNumber),-1.000);
}
else{
   fNumber=fixedstr(sNumber);
}
```

Wenn ein negatives Vorzeichen existiert, wird die aus dem String umgewandelte Zahl negiert. Anderenfalls wird nur der String in eine Festkommazahl umgewandelt.

Gehört zu: fixed.inc

Siehe auch: strtofix

8.10.65. fixed:fmul

`fixed:fmul( fixed:oper1, fixed:oper2 );`

`fixed:oper1`
Typ: Fixed (-2147482 - 2147482)

`fixed:oper2`
Typ: Fixed (-2147482 - 2147482)

Die Funktion fmul führt eine Multiplikation zweier Zahlen durch. Theoretisch steht auch „*“ als Operatorzeichen zur Verfügung. Die Erkennung durch den Compiler ist leider nicht zuverlässig, so dass die Funktion fmul empfohlen wird.

Beispiel aus math.inc (Funktion: f_abs):

```
stock fixed:f_abs(fixed:fNum) {
   if(fNum >= 0.000) {
      return fNum;
   }
   return fmul(fNum,-1.000);
}
```

Ist die Zahl positiv, wird sie unbearbeitet zurückgegeben. Anderenfalls wird sie mit -1 multipliziert (also negiert), so dass stets der Absolutwert zurückgegeben wird.

Gehört zu: fixed.inc

Siehe auch: fdiv

8.10.66. fixed:strtofix

`fixed:strtofix( sNumber[] );`

`string`
Typ: String (20)

Die Funktion strtofix wandelt einen String in eine Festkommazahl um. Im Gegensatz zu fixedstr wird ein mögliches Vorzeichen berücksichtigt.

Beispiel:

```
new sNumber[MAX_NUMBER_LENGTH];
strcpy(sNumber,"-1.378",MAX_NUMBER_LENGTH);
fNumber = strtofix(sNumber);
```

Es wird ein String definiert und anschließend mittels der Funktion strcpy mit „-1.378“ befüllt. Aus diesem String (sNumber) wird eine Festkommazahl gebildet (fNumber).

Gehört zu: math.inc

Siehe auch: fixedstr, fixtostr

8.10.67. fixtostr

`fixtostr( fixed:fNumber, sNumber[], iMaxLength );`

`fixed:fNumber`
Typ: Fixed (-2147482 - 2147482)

`string`
Typ: String (20)

`iMaxLength`
Typ: Integer (0 - 20)

Die Funktion fixtostr wandelt eine Festkommazahl in einen String um.

Beispiel aus plugin_sdal_logd_hp50[9] (Funktion: display_victim):

```
    if(g_display_distance){
        new Meters[MAX_NUMBER_LENGTH];
        fixtostr(fixed:g_PlayerEnemyDistance[UserID],Meters,MAX_NUMBER_LENGTH);
```

Wenn die Entfernung zum Gegner nach dem eigenen Ableben angezeigt werden soll, wird zunächst die gespeicherte Entfernung (g_PlayerEnemyDistance[UserID]) in einen String umgewandelt (Meters). Die Textausgabe erfolgt später im Code.

Gehört zu: math.inc

Siehe auch: fixedstr, strtofix

[9] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.68. format_command

`format_command( sUser[],sCommand[],sData[],sText[] );`

`sUser[]`
Typ: String (33)

`sCommand[]`
Typ: String (30)

`sData[]`
Typ: String (200)

`sText[]`
Typ: String (200)

Die Funktion bereitet einen Text zur Darstellung im Chat oder in den Logdateien auf, wenn ein Admin Mod Befehl ausgeführt wird. Dazu muss der ausführende Admin (sUser), der ausgeführte Befehl (sCommand) und die Parameter (sData) angegeben werden. Die Rückgabe erfolgt über sText.

Die Funktion wird üblicherweise nicht allein eingesetzt, da sie eine Unterfunktion von log_command und say_command ist.

Beispiel aus adminlib.inc (Funktion: format_command):

```
stock log_command(sUser[],sCommand[],sData[]) {
   new sText[MAX_TEXT_LENGTH];
   format_command(sUser,sCommand,sData,sText);
   log(sText);
}
```

Username (sUser), der aufgerufende Befehl (sCommand) und die zugehörigen Parameter (sData) werden an format_command übergeben, der formatierte Text (sText) geloggt.

Gehört zu: adminlib.inc

Siehe auch: log_command, say_command

8.10.69. fround

`fround( fixed:value, fround_method:method=fround_round );`

`fixed:value`
Typ: Fixed (-2147482 - 2147482)

`fround_method:method=fround_round`
Typ: Enum (0=fround_round, fround_floor, fround_ceil)

Diese Funktion rundet eine Festkommazahl. Sie wird als Integer ausgegeben. Es gibt drei verschiedene Methoden zum Runden. Die Methode fround_round rundet ab 0,5 auf und darunter ab, bei fround_floor wird grundsätzlich abgerundet, bei fround_ceil wird aufgerundet.

Beispiel aus math.inc (Funktion: fixtostr):

```
    if(fNumber<0.000){
       iNumber=fround(fNumber,fround_ceil);
    }
    else{
       iNumber=fround(fNumber,fround_floor);
    }
```

Wenn die Zahl fNumber negativ ist, wird aufgerundet, wenn sie positiv ist, wird abgerundet.

Gehört zu: fixed.inc

8.10.70. funcidx

`funcidx( const name[] );`

`const name[]`
Typ: String (20)

Die Funktion liefert den interen Index einer öffentlichen Funktion. Wird die angegebene Funktion nicht gefunden, gibt funcidx -1 zurück.

Ein sinnvoller Einsatz unter Admin Mod ist nicht gegeben.

Beispiel:
```
iID=funcidx("admin_ban");
```

Im Beispiel wird die interne ID der Funktion admin_ban aus dem Base Plugin ausgegeben.

Gehört zu: core.inc

8.10.71. get_serverinfo

`get_serverinfo( sKey[], sValue[], iMaxLength );`

`sKey[]`
Typ: String (100)

`sValue[]`
Typ: String (200)

`iMaxLength`
Typ: Integer (0 - 200)

Die Funktion liest die Daten (sValue), die mit dem Schlüssel (sKey) hinterlegt wurden, aus. Der Schlüssel und die Daten werden mit set_serverinfo gesetzt. Die Daten werden beim Mapwechsel nicht gelöscht. Der Speicher ist jedoch nur sehr gering bemessen, so dass die Verwendung dieser Funktion vermieden werden sollte.

Beispiel aus plugin_bk_cron[10] (Funktion: lag_check):

```
    get_serverinfo("last_cron",slastcheck,3);
    lastcheck=strtonum(slastcheck);
```

Der Inhalt des Schlüssels (last_cron) wird ausgelesen und mit einer maximalen Länge von 3 in die Variable slastcheck geschrieben. Der String wird anschließend in eine Ganzzahl umgewandelt. Auf diese Art und Weise kann insbesondere überprüft werden, wie lang der letzte Durchlauf vor dem Mapwechsel her ist. Beim Serverstart ist die Variable noch nicht gesetzt, so dass 0 zurückgegeben wird. Die Nutzung der vault.ini verbietet sich, da der Wert beim Beenden bzw. Absturz des Servers nicht automatisch auf 0 zurückgesetzt wird.

Gehört zu: admin.inc

Siehe auch: set_serverinfo

[10] http://www.adminmod.de/plugins.php?plugin=plugin_bk_cron

8.10.72. get_timer

`get_timer( iTimer );`

`iTimer`
Typ: Integer (0 - 511)

Die Funktion get_timer gibt den Status eines Timers aus (läuft (1), läuft nicht (0), gehört nicht dem Plugin (-1))

Beispiel:

```
if (get_timer(iTimer)) {
   kill_timer(iTimer);
}
```

Nur wenn der Timer mit dem Index (iTimer) aktuell läuft und dem eigenen Plugin gehört, wird der Timer gestoppt (kill_timer).

Gehört zu: admin.inc

Siehe auch: kill_timer, set_timer

8.10.73. get_userArmor

`get_userArmor( sPlayer[], &armor );`

`sPlayer[]`
Typ: String (33)

`&armor`
Typ: Integer (0 - 2147483647)

Die Funktion liefert den momentanen Wert der Rüstung des Spielers als Integerwert (armor) zurück.

Beispiel aus plugin_sdal_logd_hp50[11] (Funktion: hp_kill):

```
    playerinfo(iAID,Attacker,MAX_NAME_LENGTH,_,_,iATeam);
    playerinfo(iVID,Victim,MAX_NAME_LENGTH,_,_,iVTeam);

    get_userHealth(Attacker, iAHealth);
    get_userArmor(Attacker, iAArmor);
```

Die Spielernamen des Angreifers und des Opfers sowie deren Teamzugehörigkeit werden über playerinfo ermittelt. Nicht benötigte Variablen werden durch Unterstriche von der Verarbeitung ausgeklammert. Diese Methode sollte nur auf Admin Mod 2.60.42 und neuer angewendet werden, da es schon einmal Probleme mit Memory Leaks gegeben hat.

Anschließend werden die verbleibenden Lebens- und Rüstungspunkte des Angreifers zur späteren Anzeige beim Opfer ausgelesen.

Gehört zu: admin.inc

Siehe auch: get_userFrags, get_userHealth

[11] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.74. get_userAuthID

`get_userAuthID( sPlayer[], sAuthid[], iMaxLength = MAX_AUTHID_LENGTH );`

`sPlayer[]`
Typ: String (33)

`sAuthid[]`
Typ: String (39)

`iMaxLength = MAX_AUTHID_LENGTH`
Typ: Integer (0 - 39)

Die Funktion liefert die Steam ID des angegebenen Spielers (sPlayer) und speichert diese als String (sAuthid). Sollte es sich um einen Bot handeln, dann wird BOT als AuthID zurückgeliefert. Die Länge des Strings muss nicht angegeben werden, bzw. sollte stets gleich MAX_AUTHID_LENGTH (39) sein.

Beispiel aus plugin_base.inc (Funktion: admin_vote_kick):

```
            get_userAuthID(real_user,sAuthID);
            if (vote(Text,"Yes","No","HandleKickVote",sAuthID)) {
               g_VoteInProgress = 1;
               g_AbortVote = 0;
            }
```

Es wird die AuthID (i.d.R. die Steam ID) eines Spielers (real_user) ermittelt. Ein Vote zum Kicken des Spielers wird angestoßen, wobei der späteren Voteauswertung die AuthID mitgeliefert wird. Des Weiteren werden 2 globale Variablen gesetzt, wenn der Vote erfolgreich abgesetzt wurde.

Gehört zu: admin.inc

Siehe auch: get_userWONID, playerinfo

8.10.75. get_userFrags

`get_userFrags( sPlayer[], &frags );`

`sPlayer[]`
Typ: String (33)

`&frags`
Typ: Integer (0 - 2147483647)

Die Funktion liefert die Anzahl der Frags (frags) des Spieler (sPlayer) über die gesamte Mapzeit als Integerwert.

Beispiel aus plugin_sdal_logd_hp50[12] (Funktion: best_player):

```
994     for(j=1;j<=maxplayers;j++){
995        if(playerinfo(j,Player,MAX_NAME_LENGTH)){
996           get_userFrags(Player,PlayerPoints[j]);
997           if(PlayerPoints[j]>mostfrags){
998              mostfrags=PlayerPoints[j];
999           }
1000       }
1001    }
```

Mittels einer For-Schleife soll der Spieler mit den meisten Frags ermittelt werden. Die For-Schleife überprüft alle Slots. Nur Slots, die mit Spielern bestückt sind, werden ausgewertet (Zeile 995). Wurde ein Spieler gefunden, werden seine Frags (get_userFrags) ermittelt. Falls die Anzahl seiner Frags größer als der bisherige Bestwert sind, wird die Anzahl seiner Frags zum neuen Bestwert.

Gehört zu: admin.inc

Siehe auch: get_userArmor, get_userHealth

[12] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.76. get_userHealth

`get_userHealth( sPlayer[], &health );`

`sPlayer[]`
Typ: String (33)

`&health`
Typ: Integer (0 - 2147483647)

Die Funktion liefert die momentanen Lebenspunkte des Spielers als Integerwert.

Beispiel aus plugin_sdal_logd_hp50[13] (Funktion: hp_kill):

```
    playerinfo(iAID,Attacker,MAX_NAME_LENGTH,_,_,iATeam);
    playerinfo(iVID,Victim,MAX_NAME_LENGTH,_,_,iVTeam);

    get_userHealth(Attacker, iAHealth);
    get_userArmor(Attacker, iAArmor);
```

Die Spielernamen des Angreifers und des Opfers sowie deren Teamzugehörigkeit werden über playerinfo ermittelt. Nicht benötigte Variablen werden durch Unterstriche von der Verarbeitung ausgeklammert. Diese Methode sollte nur auf Admin Mod 2.60.42 und neuer angewendet werden, da es schon einmal Probleme mit Memory Leaks gegeben hat.

Anschließend werden die verbleibenden Lebens- und Rüstungspunkte des Angreifers zur späteren Anzeige beim Opfer ausgelesen.

Gehört zu: admin.inc

Siehe auch: get_userArmor, get_userFrags

[13] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.77. get_userindex

`get_userindex( sPlayer[], &iIndex );`

`sPlayer[]`
Typ: String (33)

`&iIndex`
Typ: Integer (0 - 32)

Die Funktion liefert auf Basis des Spielernamens seinen UserIndex (Server-Slot) als Integerwert.

Beispiel aus plugin_retribution (Funktion: admin_bury):

```
        get_userindex(Data, nIndex);
        playerinfo(nIndex, TargetName, MAX_NAME_LENGTH, _, _, _, nDead);
```

Der Status eines Spielers (lebend oder tot) lässt sich mit Admin Mod nur mit der Funktion playerinfo ermitteln. Da sie den Userindex benötigt, muss zunächst aus dem Namen (Data) der Userindex (nIndex) ermittelt werden. Anschließend kann der Status abgefragt werden. TargetName ist eigentlich nicht notwendig, ist jedoch eine Pflichtrückgabe. Auf die weiteren Variablen kann verzichtet werden. Als Auslassungszeichen wird der Unterstrich verwendet.

Gehört zu: admin.inc

Siehe auch: get_userSessionID, playerinfo

8.10.78. get_userinfo

`get_userinfo( sPlayer[], sKey[], sInfo[], iMaxLength );`

`sPlayer[]`
Typ: String (33) max. Länge: 100

`sKey[]`
Typ: String (100)

`sInfo[]`
Typ: String (200)

`iMaxLength`
Typ: Integer (0 - 200)

Mit der Funktion get_userinfo lassen sich einige wenige Usereinstellungen abfragen. Diese Daten werden oftmals mit setinfo angelegt. Die Funktion benötigt einen Schlüssel (sKey), dessen Daten (sInfo) ausgelesen werden können. Um mögliche Schlüssel zu ermitteln, kann man den Serverbefehl `user <username / Session ID>` auf einen bestimmten Spieler anwenden.

Der für diese Usereinstellungen zur Verfügung gestellte Speicher ist extrem klein. Statsme schaffte es in der Vergangenheit mit seinen Usereinstellungen die Admin Mod Anmeldung unmöglich zu machen. Die Fehlermeldung „info string length exceeded“ zeigte dann, dass der geringe Speicher gänzlich aufgebraucht war.

Man kann mit der Funktion execclient und dem Setzen von setinfo Usereinstellungen verankern, man sollte dies aber unter allen Umständen vermeiden, sofern es nicht einer übergeordneten Aufgabe zuträglich ist. Mehr zu diesem Thema ist dem Abschnitt Was macht eigentlich Setinfo? zu entnehmen.

Beispiel aus plugin_bk_hltvannounce[14] (Funktion: a_hltv):

```
        get_userinfo(hltvname,hspecs,hltvspecs,MAX_NAME_LENGTH);
        get_userinfo(hltvname,hslots,hltvslots,MAX_NAME_LENGTH);
        get_userinfo(hltvname,hdelay,hltvdelay,MAX_NAME_LENGTH);
        get_userIP(hltvname,hltvip,MAX_IP_LENGTH,iPort);
```

Das erste get_userinfo ruft die belegten Slots des HLTV-Servers, das zweite die maximale Slotzahl und das dritte den eingestellten Delay in Sekunden ab. Die Variablennamen der Schlüsselnamen stimmen mit ihrem Inhalt überein (hspecs z.B. hat den Inhalt „hspecs“). Anschließend wird noch die IP-Adresse des HLTVs ermittelt (get_userIP).

Gehört zu: admin.inc

[14] http://www.adminmod.de/plugins.php?plugin=plugin_bk_hltvannounce

8.10.79. get_userIP

`get_userIP( sPlayer[], sIP[], iMaxLength, &iPort = 0 );`

`sPlayer[]`
Typ: String (33)

`sIP[]`
Typ: String (22)

`iMaxLength` Typ: Integer (0 - 22)

`&iPort = 0` Typ: Integer (0 - 65535)

Diese Funktion liefert die IP des Spielers als String (sIP) auf Basis des Namens (sPlayer). Neben der IP wird auch der genutzte Port (iPort) ausgegeben.

Beispiel aus plugin_bk_hltvannounce[15] (Funktion: hltvconnect):

```
    get_userIP(hltvname,hltvip,MAX_IP_LENGTH,iPort);
    get_userinfo(hltvname,hslots,hltvinfo,MAX_NAME_LENGTH);
    ihltv=strtonum(hltvinfo);
    get_userinfo(hltvname,hspecs,hltvinfo,MAX_NAME_LENGTH);
    ihltv-=strtonum(hltvinfo);
```

Zunächst wird die IP des HLTV-Servers ermittelt. Anschließend werden die genutzte und die maximale Slotzahl ermittelt. Aus der Subtraktion ergibt sich die Anzahl der freien Plätze auf dem HLTV-Server.

Gehört zu: admin.inc

[15] http://www.adminmod.de/plugins.php?plugin=plugin_bk_hltvannounce

8.10.80. get_username

`get_username( sPlayer[], sName[], iMaxLength );`

`sPlayer[]`
Typ: String (33)

`sName[]`
Typ: String (33)

`iMaxLength`
Typ: Integer (0 - 33)

Mit dieser Funktion kann aus einem Teil des Spielernamens der gesamte Name ermittelt werden, sofern der Teil des Namens eindeutig ist. D.h. der Teil darf nicht gleichzeitig Teil des Namens eines anderen Spielers sein. Statt des Teilnamens kann auch die IP, Steam ID oder Session ID angegeben werden.

Beispiel aus plugin_base (Funktion: admin_vote_kick):

```
        get_username(Data,real_user,MAX_NAME_LENGTH);
        say_command(User,Command,Data);
        if(check_immunity(real_user)!=0) {
```

Aus Data (ob Teil-/Name, IP oder ID) wird der vollständige Spielername ermittelt. Den Einstellungen admin_quiet folgend wird die Ausführung des Befehls im Chat kommentiert (say_command). Anschließend wird über die Funktion check_immunity überprüft, ob einer Spieler dieses Namens Immunitätsrechte besitzt.

Gehört zu: admin.inc

Siehe auch: playerinfo, get_userSessionID

8.10.81. get_userorigin

`get_userorigin( sPlayer[], &iX, &iY, &iZ );`

`sPlayer[]`
Typ: String (33)

`&iX`
Typ: Integer (-2147483648 - 2147483647)

`&iY`
Typ: Integer (-2147483648 - 2147483647)

`&iZ`
Typ: Integer (-2147483648 - 2147483647)

Die Funktion gibt die Position des angegebenen Spielers (sPlayer) auf der Map zurück. Die Rückgabe erfolgt in kartesischen Koordinaten (x, y, z).

Beispiel aus plugin_retribution (Funktion: admin_bury):

```
            get_userorigin(TargetName, x, y, z);
            teleport(TargetName, x, y, (z-25));
```

Zunächst werden die Koordinaten des Spielers (TargetName) ermittelt. Anschließend wird er um 25 Einheiten tiefer gesetzt (teleport). Sofern sich der Spieler zu diesem Zeitpunkt nicht hoch in der Luft befindet, wird er in den Boden versetzt, so dass er sich nicht mehr bewegen kann.

Gehört zu: admin.inc

Siehe auch: distance, teleport

8.10.82. get_userSessionID

`get_userSessionID( sPlayer[], &iSessionID );`

`sPlayer[]`
Typ: String (33)

`&iSessionID`
Typ: Integer (0 - 2147483647)

Liefert die Session ID des Spielers als Integerwert zurück. Die Session ID ist eindeutig und wird während der gesamten Laufzeit des Servers nur einmal vergeben. Die Nutzung der Session ID für Plugins ist eher ungewöhnlich, da sie nur über get_username in einen Namen umgewandelt werden kann.

Beispiel aus plugin_spooge_AR[16] (Funktion: HandleSay):

```
        new SessionID;
        get_userSessionID (User,SessionID);
        numtostr(SessionID,CmdBuffer);
        plugin_exec("admin_slap",CmdBuffer);
```

Die Session ID wird aus dem Spielernamen (User) ermittelt und anschließend von einer Ganzzahl in einen String umgewandelt (numtostr). Abschließend wird die Session ID an den Admin Mod Befehl admin_slap weitergeleitet (plugin_exec)

Gehört zu: admin.inc

Siehe auch: get_username

[16] http://www.adminmod.de/plugins.php?plugin=plugin_spooge_AR

8.10.83. get_userTeam

`get_userTeam( sPlayer[], &team );`

`sPlayer[]`
Typ: String (33)

`&team`
Typ: Integer (0 - 4, 500, 600)

Die Funktion ermittelt die Team ID des Spielers (sPlayer). Die Teamzugehörigkeit kann auch über die Funktion playerinfo ermittelt werden.

Sonderteams sind kein Team (0), Spectator (500) und HLTV (600). Verschiedene Modifikationen haben unterschiedliche Teams:

Mod	Team 1	Team 2	Team 3	Team 4
CS	Terrorist	Counter-Terrorist	-	VIP
TFC	Team Blau	Team Rot	Team Gelb	Team Grün
DoD	Axis	Allies	-	-

Beispiel aus plugin_CS (Funktion: menuselect):

```
new Team;
get_userTeam(UserName,Team);
Team = clamp(Team,1,2) - 1;
```

Zunächst wird das Team des Spielers ermittelt. Anschließend wird zur Vermeidung einer Feldadressierung außerhalb des gültigen Bereichs (4 - AMX_ERR_BOUNDS) die möglichen Teams auf 1 und 2 begrenzt. Zur weiteren Bearbeitung werden die Teamnummern dekrementiert.

Gehört zu: admin.inc

Siehe auch: playerinfo

8.10.84. get_userWONID

`get_userWONID( sPlayer[], &iWONID );`

`sPlayer[]`
Typ: String (33)

`&iWONID`
Typ: Integer (0 - 2147483647)

Die Funktion gibt die WONID (iWONID) des Spielers (sPlayer) als Integerwert zurück. Da das WON-System von Valve[17] inzwischen durch Steam[18] ersetzt wurde, ist die Funktion nutzlos geworden. Statt dessen sollte get_userAuthID genutzt werden.

Beispiel aus plugin_sdal_look[19] (Funktion: ann_timer):

```
    get_userWONID( Player, iWon );

    if(iWon!=0){
       numtostr(iWon,strWON);
```

Auf Basis des Spielernamens (Player) wird seine WONID (iWon) ermittelt. Falls die WONID nicht 0 ist, wird sie in einen String umgewandelt.

Gehört zu: admin.inc

Siehe auch: get_userAuthID, playerinfo

[17] http://www.valvesoftware.com
[18] http://store.steampowered.com/about/
[19] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_look

8.10.85. get_vaultdata

`get_vaultdata( sKey[], sData[], iMaxLength );`

`sKey[]`
Typ: String (100)

`sData[]`
Typ: String (200)

`iMaxLength`
Typ: Integer (0 - 200)

Die Funktion liest die Daten, die unter dem Schlüssel (sKey) stehen, als String (sData) aus der vault.ini aus. Die Daten bleiben über einen Mapchange bzw. einen Serverneustart erhalten. Das Schlüssel-Daten-Paar wird mit set_vaultdata oder set_vaultnumdata angelegt. Die Funktion ist ideal um pluginspezifische Einstellungen zu speichern.

Weiß man bereits, dass die ausgelesenen Daten als Ganzzahl vorliegen, sollte man der Funktion get_vaultnumdata den Vorzug geben.

Beispiel aus plugin_retribution (Funktion: AddUserFlag):

```
if(get_vaultdata(sAuthID,VaultData,MAX_DATA_LENGTH) != 0) {
   if (strcasestr(VaultData," llama") == -1) {
      strcat(VaultData," llama", MAX_DATA_LENGTH);
      set_vaultdata(sAuthID,VaultData);
   }
```

Es wird versucht den Schlüssel mit der Steam ID des Spielers zu finden. Ist dies der Fall, wird das Ergebnis in VaultData zwischengespeichert. Falls das Ergebnis nicht bereits den Teilstring „ llama“ beinhaltet, wird dies dem Ergebis angehängt (strcat) und zurück in die vault.ini geschrieben.

Gehört zu: admin.inc

Siehe auch: get_vaultnumdata, set_vaultdata, set_vaultnumdata

8.10.86. get_vaultnumdata

`get_vaultnumdata( sKey[], &iData );`

`sKey[]`
Typ: String (100)

`&iData`
Typ: Integer (-2147483648 - 2147483647)

Die Funktion liest die Daten, die unter dem Schlüssel (sKey) stehen, als Ganzzahl (iData) aus der vault.ini aus. Die Daten bleiben über einen Mapchange bzw. einen Serverneustart erhalten. Das Schlüssel-Daten-Paar wird mit set_vaultdata oder set_vaultnumdata angelegt. Die Funktion ist ideal um pluginspezifische Einstellungen zu speichern.

Ist im Schlüssel ein String hinterlegt, darf man die Funktion nicht anwenden. Statt dessen ist get_vaultdata zu verwenden.

Beispiel aus plugin_bk_botmanager[20] (Funktion: addbot):

```
  if(get_vaultnumdata("BK_BM_BOTS",iBots) && iBots<=playercount()){
     return PLUGIN_CONTINUE;
  }
```

Der Schlüssel BK_BM_BOTS wird ausgelesen und die Daten in iBots geschrieben. Wenn der Schlüssel vorhanden und iBots kleiner gleich der Spieleranzahl ist, wird die Ausführung der Funktion addbot gestoppt und somit kein weiterer Bot hinzugefügt.

Gehört zu: admin.inc

Siehe auch: get_vaultdata, set_vaultdata, set_vaultnumdata

[20] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.87. getarg

`getarg( arg, index=0 );`

`arg`
Typ: Integer
Wertebreich: (0 - 2147483647)

`index=0`
Typ: Integer (0 - 2147483647)

Diese Funktion wird nur Fällen gebraucht, wenn man die Argumente einer Funktion mit variabler Argumentanzahl auslesen möchte. Dabei gibt „arg" die Position des Arguments beginnend mit 0 an. Das Argument „index" wird nur benötigt, wenn arg ein Feld (z.B. ein String) ist. Es gibt die auszulesende Zelle des Feldes an.

Beispiel:

```
sum(...){
   new result = 0 ;
   for (new i = 0; i < numargs(); ++i) {
      result += getarg(i);
   }
   return result;
}
```

Eine Funktion „sum" mit variabler Argumentanzahl (...) wird definiert. Eine For-Schleife fragt die einzelnen Argumente ab und addiert sie. Die Summe wird zurückgegeben. Die Anzahl der übergebenen Argumente wird mit numargs ermittelt.

Gehört zu: core.inc

Siehe auch: numargs, setarg

8.10.88. getchar

`getchar( echo=true );`

`echo=true`
Typ: Integer (0 - 1)

Diese Funktion liest die Tastatureingabe eines Zeichens aus und schreibt es bei der Option 1 in die Befehlszeile zurück. Mangels einer solchen Befehlszeile ist die Funktion in Admin Mod nutzlos.

Gehört zu: console.inc

Siehe auch: getstring, getvalue

8.10.89. getproperty

`getproperty( id=0, const name[]="", value=cellmin, string[]="" );`

`id=0`
Typ: Integer (-2147483648 - 2147483647)

`const name[]=""`
Typ: String (variabel)

`value=cellmin`
Typ: Integer (-2147483648 - 2147483647)

`const string[]=""`
Typ: String (variabel)

Mit dieser Funktion kann eine sogenannte Property bei gegebenem Schlüssel (name oder value) ausgelesen werden. Die Property wird als String im Speicher abgelegt. Es handelt sich um eine Funktion, deren Benutzung unter Admin Mod vermieden werden sollte. Stattdessen wird empfohlen z.B. auf die Funktionen get_vaultdata oder set_vaultdata zurückzugreifen.

Mehr Informationen zum Thema sind im Abschnitt 8.14 nachzulesen.

Beispiel:

```
getproperty(2,"test_prop",sString);
getproperty(3,15,sString);
```

Das erste Beispiel liest die in ID 2 befindliche Property „test_prop“ in die Variable sString aus, während das zweite Beispiel die in ID 3 befindliche Property 15 ausliest.

Gehört zu: core.inc

Siehe auch: existproperty, deleteproperty, setproperty

8.10.90. getstring

`getstring( string[], size=sizeof string, bool:pack=false );`

`string`
Typ: String (200)

`size=sizeof string`
Typ: Integer (0 - 200)

`bool:pack=false`
Typ: Integer (0 - 1)

Diese Funktion liest die Tastatureingabe komplett als String aus. Mit bool:pack kann definiert werden, ob der String gepackt oder ungepackt erwartet und ausgelesen wird. Mangels einer Befehlszeile für diese Tastatureingabe ist die Funktion in Admin Mod nutzlos.

Gehört zu: console.inc

Siehe auch: getchar, getvalue

8.10.91. getstrvar

`getstrvar( sVarname[], sValue[], iMaxLength );`

`sVarname[]`
Typ: String (100)

`sValue[]`
Typ: String (200)

`iMaxLength`
Typ: Integer (0 - 200)

Mit der Funktion kann man eine Servervariable (sVarname) als String (sValue) auslesen.

Beispiel aus plugin_base (Funktion: admin_pass):

```
        new sPasswd[MAX_NAME_LENGTH];
        getstrvar( "sv_password", sPasswd, MAX_NAME_LENGTH );
        snprintf(Text, MAX_TEXT_LENGTH, "The password is: %s.", sPasswd);
        selfmessage(Text);
```

Die Servervariable „sv_password“ wird in sPasswd ausgelesen, formatiert (snprintf) und anschließend beim Spieler, der den Befehl ausgeführt hat, ausgegeben.

Gehört zu: admin.inc

Siehe auch: exec, getvar, setstrvar

8.10.92. gettarget

`gettarget( sPlayer[], sTargetName[], iMaxLength, iRange = 2048 );`

`sPlayer[]`
Typ: String (33)

`sTargetName[]`
Typ: String (33)

`iMaxLength`
Typ: Integer (0 - 33)

`iRange = 2048`
Typ: Integer (0 - 2147483647)

Mit dieser Funktion kann man ermitteln, auf welchen Gegner (sTargetName) der Spieler (sPlayer) zielt. Sie gibt auch den Userindex des Spielers direkt zurück. Das Argument iRange definiert die Blickweite.

Beispiel aus plugin_blatt_monster[21] (Funktion: BBMonsterSpawn):

```
  if (strlen(TargetName) ==0) {
     gettarget(UserName,TargetName,MAX_DATA_LENGTH);
     if (strlen(TargetName) ==0) {
       return PLUGIN_HANDLED;
     }
   }
```

Wenn kein Spieler angegeben wurde, wird überprüft, auf welchen Spieler der Admin schaut. Schaut er niemanden an, bricht das Plugin die Ausführung ab.

Gehört zu: admin.inc

Siehe auch: pointto

[21] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_monster

8.10.93. getteamcount

`getteamcount( iTeam );`

`&team`
Typ: Integer (0 - 4, 500, 600)

Die Funktion ermittelt die Anzahl der Spieler eines Teams auf Basis der Team ID (iTeam).

Sonderteams sind kein Team (0), Spectator (500) und HLTV (600). Verschiedene Modifikationen haben unterschiedliche Teams:

Mod	Team 1	Team 2	Team 3	Team 4
CS	Terrorist	Counter-Terrorist	-	VIP
TFC	Team Blau	Team Rot	Team Gelb	Team Grün
DoD	Axis	Allies	-	-

Beispiel aus plugin_bk_botmanager[22] (Funktion: addbot):

```
    new iTplayers=getteamcount(TERROR);
    new iCTplayers=getteamcount(CT);
```

Die Variablen iTplayers und iCTplayers werden deklariert und gleichzeitig mit der Spielerzahl des jeweiligen Teams gefüllt. Dabei sind TERROR und CT jeweils Konstanten (1 bzw. 2).

Gehört zu: admin.inc

Siehe auch: get_userTeam, playerinfo

8.10.94. getvalue

`getvalue( base=10, end='^r', ... );`

`base`
Typ: Integer (2 - 32)

`end`
Typ: String (1)

Diese Funktion liest die Tastatureingabe als Ganzzahl aus. Dabei kann die Basis (base) definiert werden, sowie Abschlusszeichen für die Eingabe. Mangels einer Befehlszeile für diese Tastatureingabe ist die Funktion in Admin Mod nutzlos.

Gehört zu: console.inc

[22] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.95. getvar

`getvar( sVarname[] );`

`sVarname[]`
Typ: String (100)

Mit der Funktion kann man eine Servervariable (sVarname) als Ganzzahl auslesen.

Beispiel aus plugin_CS (Funktion: HandleKickVote):

```
      new Ratio = getvar("kick_ratio");
      if (VoteCount >= Ratio*UserCount/100) {
         if (g_AbortVote) {
            say("Kick vote was aborted by an admin");
         } else {
            set_timer("KickUser",10,1,VoteUser);
         }
```

Die Servervariable „kick_ratio“ wird als Ganzzahl in Ratio ausgelesen. Falls genügend Stimmen für den Kick zusammengekommen sind, wird entweder der Kick abgebrochen, weil ein Admin ihn aufgehalten hat, oder nach 10 Sekunden der entsprechende Spieler vom Server geworfen.

Gehört zu: admin.inc

Siehe auch: getstrvar

8.10.96. glow

`glow( sTarget[], iRed = 0, iGreen = 0, iBlue = 0 );`

`sTarget[]`
Typ: String (33)

`iRed = 0`
Typ: Integer (0 - 255)

`iGreen = 0`
Typ: Integer (0 - 255)

`iBlue = 0`
Typ: Integer (0 - 255)

Die Funktion lässt den Spieler sTarget in einer Farbe leuchten. Die Farbanteile werden über RGB (Rot, Grün, Blau) angegeben. Das Glühen wird abgeschaltet, indem alle Farbwerte auf 0 gesetzt werden.

Beispiel aus plugin_fun (Funktion: GlowHelper):

```
GlowHelper(User[], Color[]) {
   new iGoodColor = 1;

   if (streq(Color,"red")==1) {
      glow(User,250,10,10);
   } else if ( streq(Color, "blue")==1) {
      glow(User,10,10,250);
   } else if ( streq(Color, "green")==1) {
      glow(User,10,250,10);
   } else if ( streq(Color, "white")==1) {
      glow(User,250,250,250);
   } else if ( streq(Color, "yellow")==1) {
      glow(User,250,250,10);
   } else if ( streq(Color, "purple")==1) {
      glow(User,250,10,250);
   } else if ( streq(Color, "off")==1) {
      glow(User,0,0,0);
   } else {
      iGoodColor = 0;
   }
   return iGoodColor;
}
```

Die Funktion GlowHelper wandelt Farben, die in einem String (Color) übergeben werden, in einen RGB-Wert um und lässt den entsprechenden Spieler (User) in dieser Farbe glühen. Bei „off" wird das Glühen abgeschaltet. Ist die übergebende Farbe nicht definiert gibt die Funktion einen Fehlerwert (iGoodColor) zurück.

Gehört zu: admin.inc

8.10.97. godmode

`godmode( sPlayer[], iOnOff );`

`sPlayer[]`
Typ: String (33)

`iOnOff`
Typ: Integer (0 - 1)

Die Funktion macht den Spieler (sPlayer) unverwundbar, wenn man als iOnOff eine 1 übergibt. Mit 0 stellt man den Godmode wieder aus. Es wird eine Nachricht allen Spielern auf dem Server angezeigt, dass der Spieler diese Fähigkeit erhalten hat. Diese Nachricht kann nicht ausgeschaltet werden!

Beispiel aus plugin_cheat (Funktion: admin_godmode):

```
    if (check_user(strGodModeUser)==1) {
       say_command(User,Command,Data,1);
       godmode(strGodModeUser,iGodMode);
    } else {
       selfmessage("Unrecognized player: ");
       selfmessage(strGodModeUser);
    }
```

Wenn der entsprechende Spieler (strGodModeUser) auf dem Server ist, wird die Verwendung des Godmodes bekannt gegeben. Aber auch ohne say_command wird eine Meldung an alle Spieler abgesetzt. Da hier erheblich in die Spielmechanik eingegriffen wird, kann die Meldung nicht unterdrückt werden. Anschließend wird der Godmode für den entsprechenden Spieler ausgeführt.

Ist der Spieler nicht auf dem Server, wird das dem aufrufenden Admin in der Console mitgeteilt.

Gehört zu: admin.inc

Siehe auch: noclip

8.10.98. heapspace

`heapspace( );`

Die Funktion gibt die Größe des freien Heap-Speichers in Zellen zurück.

Diese Funktion wird verwendet, um Speicherlöcher in Scripten aufzuspüren. Speicherlöcher sind Fehler im Programm, die den verwendeten Speicher immer weiter anwachsen lassen, bis keiner mehr zur Verfügung steht und die Scriptausführung mit dem Fehler AMX_ERR_STACKERR beendet werden muss.

Üblicherweise liegt der Fehler allerdings nicht am Script selber, sondern am Wirtsprogramm (= Adminmod). Der Stack und der Heap belegen einen gemeinsamen Speicherbereich im Script, in dem dynamische Daten abgelegt werden (z.B. Funktionsparameter, lokale Variablen). Die Größe dieses Speicherbereiches kann durch die Präprozessordirektive #pragma dynamic n verändert werden, wobei n die Größe in Zellen ist. Die Standardgröße ist 2048 Zellen.

Beispiel aus plugin_blatt_map[23] (Funktion: DebugHeap):

```
DebugHeap(context[]) {
  if (g_DebugLevel >= 2) {
    new heapsize[MAX_TEXT_LENGTH];
    snprintf(heapsize,MAX_TEXT_LENGTH,"[%s] %i free bytes.",context,heapspace());
    plugin_message(heapsize);
  }
  return 0;
}
```

Wenn der in der globalen Variable (g_DebugLevel) Debuglevel größer oder gleich zwei ist, wird der noch verbleibende Heap-Speicher ermittelt und mittels plugin_message in der Console (Spieler- oder Serverconsole) ausgegeben.

Gehört zu: core.inc

[23] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_map

8.10.99. help

```
help( sKeyword[]);
```

`sKeyword[]`
Typ: String (100)

Diese Funktion ist veraltet und wird nicht mehr in Scripts verwendet.

Wenn früher in der adminmod.cfg ein help_file gesetzt wurde, konnte man z.B. per Script `help("vote")` ausführen, um sich alle Beschreibungen zu „vote“ in der Konsole anzeigen zu lassen. Wenn kein Schlüsselwort (sKeyword) angegeben wurde, wurden alle Funktionen angezeigt.

Die Funktionalität wurde durch den direkten Admin Mod Befehl admin_help und die Funktionen plugin_registercmd und plugin_registerhelp ersetzt.

Gehört zu: admin.inc

Siehe auch: plugin_registercmd, plugin_registerhelp

8.10.100. index

`index( sSource[], iChar );`

`sSource[]`
Typ: String (200)

`iChar`
Typ: Integer (0 - 256)

Die Funktion ermittelt die Position des ersten Vorkommens des Zeichens (iChar) im String (sSource) von links beginnend. Für das Zeichen muss die ASCII-Nummer angegeben werden. Vereinfacht kann auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden. Die Funktion liefert -1 zurück, falls das Zeichen nicht gefunden wurde.

Die Funktion ist ein Alias für strchr.

Beispiel aus plugin_blatt_map[24] (Funktion: ReadMapLine):

```
    pos=index(Line,' ');
    if (pos>0) {
        Line[pos]=NULL_CHAR;
    }

```

Es wird das erste Leerzeichen von links in Line gesucht. Wurde ein Leerzeichen gefunden, wird es durch ein String-Endzeichen (NULL_CHAR) ersetzt. Alle Funktionen, die den String von links zu lesen beginnen, werden an dieser Stelle beendet. Beginnt man von rechts, werden die Zeichen hinter dem String-Endzeichen zuerst gelesen. Man sollte daher genau wissen, was man tut, wenn man ein String-Endzeichen manuell setzt. Die Verwendung von strsep wäre eine Alternative.

Gehört zu: string.inc

Siehe auch: rindex, strchr, strrchr

[24] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_map

8.10.101. kick

`kick( sPlayer[] );`

`sPlayer[]`
Typ: String (33)

Die Funktion wirft den Spieler (sPlayer) vom Server.

Beispiel aus plugin_base (Funktion: admin_kick):

```
if (strlen(Reason) != 0) {
   snprintf(Text, MAX_TEXT_LENGTH, "You have been kicked because %s", Reason);
   message(real_user, Text);
}
kick(real_user);
```

Es wird überprüft, ob ein Grund für den Kick angegeben wurde. Wenn es der Fall ist, wird dies dem Spieler mitgeteilt (message). Anschließend wird der Spieler vom Server geworfen.

Gehört zu: admin.inc

Siehe auch: ban

8.10.102. kill_timer

`kill_timer( iTimer );`

`iTimer`
Typ: Integer (0 - 512)

Mit dieser Funktion kann man einen laufenden Timer abbrechen. Dafür wird der Timer-Index (iTimer) benötigt, den man beim Ausführen des Timers set_timer zurückgeliefert bekommt. Es können nur Timer abgebrochen werden, die vom selben Plugin ausgeführt wurden.

Beispiel aus plugin_fun (Funktion: KillDisco):

```
for (i = 1; i <= iMaxPlayers; i++) {
   if(playerinfo(i,Name,MAX_NAME_LENGTH)==1) {
      glow(Name,0,0,0);
   }
}
centersay("Disco . . . Is Dead.",10,0,255,0);
kill_timer(iDiscoTimer);
```

Alle Plätze des Servers werden überprüft. Falls ein Platz durch einen Spieler belegt wird, wird ein eventuelles Glühen abgestellt. Eine zehnsekündige Nachricht in Grün verkündet, dass Disco tot ist. Anschließend wird der Discotimer mit dem Index iDiscoTimer abgebrochen.

Gehört zu: admin.inc

Siehe auch: set_timer

8.10.103. list_maps

```
list_maps( );
```

Die Funktion zeigt dem ausführenden Spieler alle Maps aus der mapcycle.txt in der Console.

Beispiel aus plugin_base (Funktion: admin_listmaps):

```
    selfmessage("The maps on the mapcycle are:");
    list_maps();
    selfmessage("and the current map is:");
    currentmap(curmap,100);
    selfmessage(curmap);
```

Es wird eine Überschrift für die zur Verfügung stehenden Maps im Cycle in der Console ausgegeben. Es folgt die Ausgabe dieser Maps und die Überschrift für die aktuelle Map. Die aktuelle Map wird ermittelt und in der Console ausgegeben.

Gehört zu: admin.inc

8.10.104. listspawn

`listspawn( sClass[] );`

`sClass[]`
Typ: String (100)

Die Funktion gibt alle Items an, die gespawned wurden. Man kann aber auch nur nach einem Teil (sClass) suchen.

Die Funktionalität steht mittlerweile nicht mehr zur Verfügung.

Beispiel aus plugin_spawn (Funktion: admin_listspawn):

```
    convert_string(HLData,Data,MAX_DATA_LENGTH);
    listspawn(Data);
```

Der Suchtext wird von einem HL- in einen Small-String konvertiert (convert_string). Anschließend werden die in das Muster fallenden Items in der Console angezeigt.

Gehört zu: admin.inc

Siehe auch: movespawn, removespawn, spawn

8.10.105. log

`log( sLogEntry[] );`

`sLogEntry[]`
Typ: String (256)

Mit dieser Funktion können Daten (sLogEntry) in die Logdateien geschrieben werden.

Beispiel aus plugin_base (Funktion: plugin_init):

```
870     currentmap(strMap, MAX_DATA_LENGTH);
871     snprintf(ExecCommand, MAX_DATA_LENGTH, "%s.cfg", strMap);
872     if ( fileexists(ExecCommand) ) {
873        snprintf(ExecCommand, MAX_DATA_LENGTH, "exec %s.cfg", strMap);
874        log(ExecCommand);
875        exec(ExecCommand);
876     }
```

Es wird die aktuelle Map ermittelt und überprüft, ob sich im Modverzeichnis eine Konfiguartionsdatei für diese spezielle Map befindet. Existiert die Datei, wird sie ausgeführt. Dies ist eine oftmals übersehene Funktion Admin Mods, die der Basisfunktionalität des HL-Servers zugeschlagen wird.

Unter anderem wird der ausgeführte Befehl in die Logdateien geschrieben (Zeile 874).

Gehört zu: admin.inc

Siehe auch: log_command

8.10.106. log_command

`log_command( sUser[],sCommand[],sData[] );`

`sUser[]`
Typ: String (33)

`sCommand[]`
Typ: String (30)

`sData[]`
Typ: String (200)

Die Funktion schreibt einen formatierten Text in die Logdateien. Dazu muss der ausführende Admin (sUser), der ausgeführte Befehl (sCommand) und die Parameter (sData) angegeben werden. Die Ausgabe kann durch die Einstellung von admin_quiet beeinflusst werden.

Beispiel aus plugin_base (Funktion: admin_ssay):

```
    strstripquotes(Data);
    say(Data);
    log_command(User,Command,Data);
```

Der in den Chat zu schreibende Text (Data) wird möglicher Anführungszeichen befreit und ausgegeben. Anschließend wird der Text inklusive Admin und dem genutzten Befehl in die Logdateien geschrieben.

Gehört zu: adminlib.inc

Siehe auch: format_command, log, say_command

8.10.107. look_in_dir

`look_in_dir( sDirectory[], sMatch[], sFilename [], iNumber );`

`sDirectory[]`
Typ: String (100)

`sMatch[]`
Typ: String (100)

`sFilename []`
Typ: String (100)

`iNumber`
Typ: Integer (0 - 2147483647)

Diese Funktion ist nur unter dem Betriebssystem Windows möglich. Unter Angabe des Verzeichnisses (sDirectory) wird überprüft, ob eine Datei unter Angabe eines Musters (sMatch) existiert. Wenn dies der Fall ist, wird der Dateiname an einen String (sFileName) übergeben. Es ist auch möglich „*" als Joker im Muster zu verwenden. Die Ausgabe des x-ten Eintrags kann über iNumber eingestellt werden.

Beispiel aus plugin_rules[25] (Funktion: showrules):

```
new i = 0;
rulefound = look_in_dir("rules", "*.cfg", strRule, i);

while (rulefound == 1 && i < 100) {
   len = strlen(strRule)-3;
   strncpy(strRule, strRule, len, len);
   selfmessage(strRule);
   i++;
   rulefound = look_in_dir("rules", "*.cfg", strRule, i);
}
```

Es wird im Verzeichnis „rules" nach allen Konfigurationsdateien („*.cfg") gesucht. Dabei wird das erste Vorkommen (i) in strRule zurückgegeben. In einer While-Schleife werden weitere Dateien ermittelt. Dabei wird vor der Ausgabe des Dateinamens dieser Suffix entfernt.

Gehört zu: admin.inc

[25] http://www.adminmod.de/plugins.php?plugin=plugin_rules

8.10.108. maptime

`maptime( iWhichtime, iPrintToConsole = 0 );`

`iWhichtime`
Typ: Integer (0 - 1)

`iPrintToConsole = 0`
Typ: Integer (0 - 1)

Die Funktion kann die abgelaufene und die verbleibende Spielzeit (in Sekunden) auf der aktuellen Map angeben. Es ist zu definieren (iWhichtime), ob die abgelaufene (0) oder die verbleibende (1) Zeit ausgegeben werden soll. Man kann auch optional angeben, ob die Zeit in die Console geschrieben werden soll.

Die Funktion erkennt nicht den Restart einer Map in Counter-Strike. In diesem Fall sollte auf timeleft oder LogD basierte Berechnungen zurückgegriffen werden.

Beispiel aus plugin_wm_chat[26] (Funktion: wmtl_check):

```
public wmtl_check() {
   new iTmp = maptime(1, 0);
   switch(iTmp) {
      case 1200: {wm_timeleft(1);}
      case 600: {wm_timeleft(1);}
      case 300: {wm_timeleft(1);}
      case 150: {wm_timeleft(1);}
      case 60: {wm_timeleft(1);}
      case 5: {execall("speak five");}
      case 4: {execall("speak four");}
      case 3: {execall("speak three");}
      case 2: {execall("speak two");}
      case 1: {
         execall("speak one");
         kill_timer(TV[0]);
         kill_timer(TV[1]);
      }
   }
}
```

Im Beispiel wird die verbleibende Mapzeit ermittelt, ohne sie in die Console zu schreiben. Über eine Switch-Verzweigung wird eine userspezifische Funktion angesprochen bzw. via HL-Speech (vox) von fünf auf eins heruntergezählt. Bei 1 werden zwei Timer abgebrochen.

Gehört zu: admin.inc

Siehe auch: timeleft

[26] http://www.adminmod.de/plugins.php?plugin=plugin_wm_chat

8.10.109. matherror

`matherror( iError );`

`iError`
Typ: Integer (0 - 3)

Wandelt einen Fehlercode aus den Festkommaberechnungen der math.inc in lesbaren Text um und schreibt diesen in die Logdateien. Kann in Kombination mit den meisten Festkommaberechnungen verwendet werden.

Beispiel:

```
new fixed:fNegative=-4.000;
new iError=0;
iError=f_sqrt(fNegative);
matherror(iError);
```

Es wird versucht aus einer negativen Zahl (fNegative), die Wurzel zu ziehen. Die Variable fNegative wird auf -1 und der Fehlercode (iError) auf 1 gesetzt. Mit matherror wird die Meldung „No valid value for function input!“ in die Logs geschrieben.

Gehört zu: math.inc

Siehe auch: f_degtorad

8.10.110. max

`max( a, b );`

`a`
Typ: Integer (-2147483648 - 2147483647)

`b`
Typ: Integer (-2147483648 - 2147483647)

Die Funktion liefert den Maximalwert zweier Ganzzahlen. Warum nicht auf die nativen Funktionen zurückgegriffen wird (core.inc), entzieht sich der Kenntnis des Autors.

Eine vergleichbare Funktion f_max gibt es auch für Festkommazahlen.

Beispiel aus plugin_bk_botmanager[27] (Funktion: admin_bot_set):

```
    if(strcasecmp(sBotCvar,"bots")==0){
        iValue=min(max(strtonum(sValue),0),maxplayercount());
        if(get_vaultnumdata("BK_BM_BOTS",iBots) && iBots<iValue){
            set_vaultnumdata("BK_BM_BOTS",iValue);
```

Wenn die Option, die übergeben wurde, gleich „bots“ ist, wird die übergebene Botanzahl zunächst durch eine Max-Auswahl mit 0 nach unten und anschließend durch eine Min-Auswahl gegen die maximale Spielerzahl (maxplayercount) begrenzt. Dies hätte man auch über die Funktion clamp lösen können.

Anschließend wird überprüft, ob der Eintrag „BK_BM_BOTS“ in der vault.ini existiert, und ob die bisherige dort abgelegte Botanzahl kleiner als der neue Wert ist. Der neue Wert wird anschließend in „BK_BM_BOTS“ geschrieben.

Gehört zu: adminlib.inc

Siehe auch: clamp, f_max, f_min, min

[27] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.111. maxplayercount

`maxplayercount( );`

Die Funktion ermittelt die maximale mögliche Spielerzahl auf dem Server.

Beispiel aus plugin_fun (Funktion: KillGlow):

```
KillGlow() {
	new i;
	new iMaxPlayers = maxplayercount();
	new Name[MAX_NAME_LENGTH];

	for (i = 1; i <= iMaxPlayers; i++) {
		if(playerinfo(i,Name,MAX_NAME_LENGTH)==1) {
			glow(Name,0,0,0);
		}
	}
}
```

Die maximale Spielerzahl wird ermittelt und als Maximalwert für die For-Schleife verwendet. Auf diese Weise muss man nicht alle 32 theoretisch möglichen Slots durchsuchen. In dieser For-Schleife wird bei allen Spielern das Glühen abgeschaltet.

Gehört zu: admin.inc

Siehe auch: playercount

8.10.112. menu

`menu( username[],text[],keys ,time=0 );`

`username[]`
Typ: String (33)

`text[]`
Typ: String (512)

`keys`
Typ: Integer (0 - 1024)

`time=0`
Typ: Integer (0 - 2147483647)

Unter Angabe des Spielers (username) kann man eine Nachricht (text) in Form eines Menüs anzeigen lassen. Mit dem der Variable keys wird definiert, welche Ziffertasten als Eingabe-Optionen verwendet werden sollen. Die Zahl errechnet sich aus der Summe der Zweierpotenzen von 2^{n-1}. Die Zahl n steht für die jeweilige Ziffertaste. Die Ziffer 0 ist für die Variable keys eine 10 ($2^{10-1} = 512$). Der Wert 1024 sperrt alle Ziffertasten.

Sollen z.B. nur die Tasten 0 bis 3 auswählbar sein, ergibt sich

$$keys = 2^{10-1} + 2^{1-1} + 2^{2-1} + 2^{3-1} = 512 + 1 + 2 + 4 = 519$$

Mit der Variable time definiert man die Dauer der Bildschirmausgabe. Wird der Wert nicht angegeben oder wird eine 0 übergeben, bleibt das Menü offen bis etwas ausgewählt wurde oder das Menü von einem anderen überschrieben wurde. Es gibt keine Überprüfungsmöglichkeit, ob beim Spieler bereits ein Menü geöffnet ist.

Die Auswertung erfolgt über den registrierten HL-Befehl „menuselect“ (Tutorial).

Beispiel aus plugin_CS (Funktion: DrawSubMenu):

```
    strcat(Menutext,"^n^n\w0. Back",512);
    Keys |= 512;
    playerinfo(UserIndex,UserName,MAX_NAME_LENGTH);
    g_UserMenuSelected[UserIndex] = -Menu;
    menu(UserName,Menutext,Keys,0);
```

Es wird an den Menütext (Menutext) die Option 0 (Back) angehängt. Ein bitweises „Oder“ wird auf Keys und 512 angesetzt und das Ergebnis in Keys geschrieben. Hier hätte man auch einfach addieren können. Mittels playerinfo wird der Spielername ermittelt und die negative Nummer des Menüs wird in das globale Feld eingetragen, damit man weiß, in welchem Menü der Spieler sich befindet. Abschließend wird das Menü permanent bis zu einer Optionsauswahl bzw. einem überschreibenden Menü angezeigt.

Gehört zu: admin.inc

Siehe auch: vote

8.10.113. message

`message( sTarget[], sMessage[] );`

`sTarget[]`
Typ: String (33)

`sMessage[]`
Typ: String (100)

Die Funktion schreibt eine Nachricht (sMessage) in die Console der angegebenen Person (sTarget). Die Funktion messageex bietet einen größeren Umfang.

Beispiel aus plugin_base (Funktion: admin_kick):

```
if (strlen(Reason) != 0) {
   snprintf(Text, MAX_TEXT_LENGTH, "You have been kicked because %s", Reason);
   message(real_user, Text);
}
kick(real_user);
```

Es wird überprüft, ob ein Grund für den Kick angegeben wurde. Wenn dies der Fall ist, wird dies dem Spieler in der Konsole mitgeteilt. Anschließend wird der Spieler vom Server geworfen (kick).

Gehört zu: admin.inc

Siehe auch: centersayex, directmessage, messageex

8.10.114. messageex

`messageex( sTarget[],sMessage[], print_type:iMessageType );`

`sTarget[]`
Typ: String (33)

`sMessage[]`
Typ: String (100)

`print_type:iMessageType`
Typ: Enum (0=print_console, print_center, print_chat, print_tty, print_pretty)

Die Funktion schreibt eine Nachricht (sMessage) an die angegebene Person (sTarget). Die Art, wie sie dem Spieler angezeigt werden soll, wird mit print_type definiert.

Es gibt 5 Darstellungen:

print_console Anzeige in der Konsole

print_center Anzeige in normaler Schrift in der Mitte des Bildschirmes

print_chat Anzeige im Chat

print_tty Anzeige in gelb am linken, unteren Bildschirmrand

print_pretty Anzeige in grün in der Mitte des Bildschirmes

Beispiel aus plugin_base (Funktion: admin_psay):

```
snprintf(Text, MAX_TEXT_LENGTH, "(Private Msg From %s): %s", User, Msg);
if (iMsgToAdmin == 1) {
   log(Text);
} else {
   messageex(TargetName, Text, print_chat);
}
```

Stellt einen Präfix mit dem Absender (User) vor die eigentliche Nachricht (Msg). Wenn die Nachricht an einen Admin in der Serverconsole gehen soll, wird der Text in die Logdateien geschrieben. Anderenfalls wird dem Empfänger (TargetName) die Nachricht im Chat präsentiert.

Gehört zu: admin.inc

Siehe auch: centersayex, directmessage, message, say, selfmessage

8.10.115. min

`min( a, b );`

`a`
Typ: Integer (-2147483648 - 2147483647)

`b`
Typ: Integer (-2147483648 - 2147483647)

Die Funktion liefert den Minimalwert zweier Ganzzahlen. Warum nicht auf die nativen Funktionen zurückgegriffen wird (core.inc), entzieht sich der Kenntnis des Autors.

Eine vergleichbare Funktion f_min gibt es auch für Festkommazahlen.

Beispiel aus plugin_bk_botmanager[28] (Funktion: admin_bot_set):

```
    if(strcasecmp(sBotCvar,"bots")==0){
       iValue=min(max(strtonum(sValue),0),maxplayercount());
       if(get_vaultnumdata("BK_BM_BOTS",iBots) && iBots<iValue){
          set_vaultnumdata("BK_BM_BOTS",iValue);
```

Wenn die Option, die übergeben wurde, gleich „bots“ ist, wird die übergebene Botanzahl zunächst durch eine Max-Auswahl mit 0 nach unten und anschließend durch eine Min-Auswahl gegen die maximale Spielerzahl (maxplayercount) begrenzt. Dies hätte man auch über die Funktion clamp lösen können.

Anschließend wird überprüft, ob der Eintrag „BK_BM_BOTS“ in der vault.ini existiert, und ob die bisherige dort abgelegte Botanzahl kleiner als der neue Wert ist. Der neue Wert wird anschließend in „BK_BM_BOTS“ geschrieben.

Gehört zu: adminlib.inc

Siehe auch: clamp , f_max, f_min, max

[28] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.116. motd

`motd( sTarget[],sMessage[] );`

`sTarget[]`
Typ: String (33)

`sMessage[]`
Typ: String (1536)

Sofern die Modifikation es unterstützt, kann man mit dieser Funktion ein „Message of the Day“ Fenster mit einer Nachricht (sMessage) bei einem bestimmten Spieler (sTarget) öffnen. Die Anzahl der Zeichen ist äußerst begrenzt. Insbesondere bei HTML-Seiten sollte man besser nur eine automatische Weiterleitung auf eine echte HTML-Seite generieren.

Beispiel aus plugin_base (Funktion: admin_motd):

```
strbreak(Data, Target, Msg, MAX_TEXT_LENGTH);
if (strlen(Msg) == 0) {
   selfmessage( "Unparsable format: no message found.");
   return PLUGIN_HANDLED;
} else if (check_user(Target) == 0) {
   selfmessage("Unrecognized player: ");
   selfmessage(Target);
   return PLUGIN_HANDLED;
}
get_username(Target,TargetName,MAX_NAME_LENGTH);

motd( TargetName, Msg );
```

Zunächst werden die übergebenen Daten aus dem Befehl in einen Spieler- und einen Nachrichtteil getrennt. Trennzeichen ist das erste Leerzeichen (strbreak). Wenn keine Nachricht gefunden wurde oder der Spieler nicht auffindbar ist, wird die Abarbeitung mit einer Fehlermeldung an den aufrufenden Admin abgebrochen. Die Funktion get_username macht aus einer ID oder einer IP einen Namen und nutzt diesen zur Ausgabe der Nachricht.

Gehört zu: admin.inc

8.10.117. movespawn

`movespawn( iIdentity, iX, iY, iZ, iXAngle, iYAngle, iZAngle );`

`iIdentity`
Typ: Integer (0 - 2147483647)

`iX`
Typ: Integer (-2147483648 - 2147483647)

`iY`
Typ: Integer (-2147483648 - 2147483647)

`iZ`
Typ: Integer (-2147483648 - 2147483647)

`iXAngle`
Typ: Integer (0 - 359)

`iYAngle`
Typ: Integer (0 - 359)

`iZAngle`
Typ: Integer (0 - 359)

Die Funktion verschiebt ein Item (iIdentity) auf eine bestimmte Position (iX, iY, iZ) und in einer bestimmten Ausrichtung (iXAngle, iYAngle, iZAngle).

Die Funktionalität steht mittlerweile nicht mehr zur Verfügung.

Beispiel aus plugin_spawn (Funktion: admin_movespawn):

```
    if(movespawn(iIdentity,X,Y,Z,XAngle,YAngle,ZAngle)==1) {
       selfmessage("Success.");
       say_command(User,Command,Data);
    } else {
       selfmessage("Failed.");
    }
```

Es wird versucht ein Item zu verschieben. Falls dies erfolgreich war, wird eine Erfolgsmeldung an den Admin und alle anderen Spieler ausgegeben. Anderenfalls wird nur der Admin über den fehlgeschlagenen Versuch informiert.

Gehört zu: admin.inc

Siehe auch: listspawn, removespawn, spawn

8.10.118. nextmap

`nextmap( sMap[], iMaxLength );`

`sMap[]`
Typ: String (100)

`iMaxLength`
Typ: Integer (0 - 100)

Die Funktion gibt die nach der aktuellen geladene Map (sMap) aus. Sie berücksichtigt keine Plugins, die in den Mapwechsel eingreifen, sondern geht in der Ermittlung nach der Methode des HL-Servers vor. Nur bei der ersten Map nach dem Serverstart stimmt die Anzeige nicht.

Beispiel aus plugin_chat (Funktion: SayNextMap):

```
SayNextMap() {
   nextmap(NextMap,MAX_NAME_LENGTH);
   snprintf(Text, MAX_TEXT_LENGTH, "The next map will be: %s", NextMap);
   say(Text);
}
```

Die nächste Map wird ermittelt, in einen Text eingebettet und im Chat an alle Spieler ausgegeben.

Gehört zu: admin.inc

Siehe auch: currentmap

8.10.119. noclip

`noclip( sPlayer[], iOnOff );`

`sPlayer[]`
Typ: String (33)

`iOnOff`
Typ: Integer (0 - 1)

Die Funktion lässt den Spieler (sPlayer) durch Wände gehen, wenn man als iOnOff eine 1 übergibt. Mit 0 stellt man den Noclip Modus wieder aus. Es wird eine Nachricht allen Spielern auf dem Server angezeigt, dass der Spieler diese Fähigkeit erhalten hat. Diese Nachricht kann nicht ausgeschaltet werden!

Beispiel aus plugin_cheat (Funktion: admin_noclip):

```
    if (check_user(strNoclipUser)==1) {
       say_command(User,Command,Data,1);
       noclip(strNoclipUser,iNoclip);
    } else {
       selfmessage("Unrecognized player: ");
       selfmessage(strNoclipUser);
    }
```

Wenn der entsprechende Spieler (strNoclipUser) auf dem Server ist, wird die Verwendung des Noclip Modus bekannt gegeben. Aber auch ohne say_command wird eine Meldung an alle Spieler abgesetzt. Da hier erheblich in die Spielmechanik eingegriffen wird, kann die Meldung nicht unterdrückt werden. Anschließend wird der Noclip Modus für den entsprechenden Spieler ausgeführt.

Ist der Spieler nicht auf dem Server wird das dem aufrufenden Admin in der Console mitgeteilt.

Gehört zu: admin.inc

8.10.120. numargs

`numargs( );`

Diese Funktion wird nur Fällen gebraucht, wenn man die Anzahl der Argumente einer Funktion mit variabler Anzahl auslesen möchte.

Beispiel:

```
sum(...){
   new result = 0 ;
   for (new i = 0; i < numargs(); ++i) {
      result += getarg(i);
   }
   return result;
}
```

Eine Funktion „sum“ mit variabler Argumentanzahl (...) wird definiert. Eine For-Schleife fragt die einzelnen Argumente ab (getarg) und addiert sie. Die Summe wird zurückgegeben. Die Anzahl der übergebenen Argumente wurde dabei mit numargs ermittelt.

Gehört zu: core.inc

Siehe auch: numargs, setarg

8.10.121. numtostr

`numtostr( num, str[] );`

`num`
Typ: Integer (-2147483648 - 2147483647)

`str[]`
Typ: String (20)

Die Funktion wandelt eine Ganzzahl (num) in einen String (str) um.

Beispiel aus plugin_CS (Funktion: HandleRestartVote):

```
	new Timelimit[MAX_NUMBER_LENGTH];
	numtostr((getvar("mp_timelimit") * 60 - timeleft() + 10) / 60,Timelimit);
	setstrvar("mp_timelimit",Timelimit);
	setstrvar("sv_restartround","10");
	say("Round restart vote succeeded.");
```

Vom aktuellen Zeitlimit auf dem Server wird die Restzeit abgezogen und 10 Sekunden addiert. Dieser Wert wird als neues Zeitlimit eingetragen. Anschließend wird ein Maprestart nach 10 Sekunden angefordert und der Restart allen Spielern mitgeteilt.

Gehört zu: adminlib.inc

Siehe auch: strtonum

8.10.122. playercount

`playercount( );`

Die Funktion gibt die Anzahl der Spieler auf dem Server als Ganzzahl zurück.

Beispiel aus plugin_logd_redirect[29] (Funktion: logd_redirect):

```
Count = maxplayercount();
Count2 = playercount();

snprintf(Text,MAX_TEXT_LENGTH,"Maxplayers = %i^nPlayerCount = %i",Count,Count2);
messageex(User,Text,print_chat);
```

Es wird die maximal mögliche (Count) und die tatsächliche Spielerzahl (Count2) ermittelt. Die Ausgabe wird formatiert und im Chat des entsprechenden Spielers ausgegeben.

Gehört zu: admin.inc

Siehe auch: maxplayercount

[29] http://www.adminmod.de/plugins.php?plugin=plugin_logd_redirect

8.10.123. playerinfo

`playerinfo( iPlayerIndex, sName[], iLength, &iUserID = 0, &iWONID = 0, &iTeam = 0, &iDead = 0, sAuthID[MAX_AUTHID_LENGTH] = "" );`

`iPlayerIndex`
Typ: Integer (1 - 32)

`sName[]`
Typ: String (33)

`iLength`
Typ: Integer (0 - 33)

`&iUserID = 0`
Typ: Integer (0 - 2147483647)

`&iWONID = 0`
Typ: Integer (0 - 2147483647)

`&iTeam = 0`
Typ: Integer (0 - 4, 500, 600)

`&iDead = 0`
Typ: Integer (0 - 1)

`sAuthID[MAX_AUTHID_LENGTH] = ""`
Typ: String (39)

Diese Funktion dient zur umfassenden Ermittlung diverser Spielerinformation. Dies sind Name (sName), Session ID (iUserID), WONID (iWONID), Teamzugehörigkeit (iTeam), Lebensstatus (iDead) und die Steam ID (sAuthID). Sie gibt darüber hinaus direkt aus, ob ein Spieler mit dem zugehörigen Userindex existiert. Optionale Argumente können mit „_“ ausgespart werden, wenn nicht sie sondern die nachfolgenden benötigt werden.

Beispiel aus plugin_retribution (Funktion: admin_bury):

```
        get_userindex(Data, nIndex);
        playerinfo(nIndex, TargetName, MAX_NAME_LENGTH, _, _, _, nDead);
```

Der Status eines Spielers (lebend oder tot) lässt sich mit Admin Mod nur mit der Funktion playerinfo ermitteln. Da sie den Userindex benötigt, muss zunächst aus dem Namen (Data) der Userindex (nIndex) ermittelt werden. Anschließend kann der Status abgefragt werden. TargetName ist eigentlich nicht notwendig, ist jedoch eine Pflichtrückgabe. Auf die weiteren Variablen kann verzichtet werden. Als Auslassungszeichen wird der Unterstrich verwendet. Die Steam ID wurde ebenfalls ausgelassen.

Gehört zu: admin.inc

Siehe auch: get_userindex, get_username, get_userSessionID, get_userWONID, get_userTeam, get_userAuthID

8.10.124. playsound

`playsound( sTarget[], sSound[] );`

`sTarget[]`
Typ: String (33)

`sSound[]`
Typ: String (100)

Die Funktion spielt eine Wave Datei (sSound) beim angegebenen Spieler (sTarget) ab. Der Spieler muss diese Datei bereits auf dem Rechner haben. Bei der Angabe der Wavedatei darf der Pfad nicht vergessen werden. Der Pfad beginnt im „sound“ Verzeichnis.

Beispiel aus plugin_retribution (Funktion: PlaySoundToAll):

```
PlaySoundToAll(sSound[]) {
   new i;
   new iMaxPlayers = maxplayercount();
   new Name[MAX_NAME_LENGTH];

   if (getvar("admin_fx") != 0) {
      for (i = 1; i <= iMaxPlayers; i++) {
         if (playerinfo(i,Name,MAX_NAME_LENGTH) != 0) {
            playsound(Name, sSound);
         }
      }
   }
}
```

Das Abspielen der Sounddatei (sSound) wird nur durchgeführt, wenn admin_fx angeschaltet ist. Anschließend werden mit einer For-Schleife alle Serverslots nach Spielern durchsucht. Wenn ein Spieler gefunden wurde, wird bei im die Sounddatei (sofern auf seinem Rechner vorhanden) abgespielt.

Gehört zu: admin.inc

Siehe auch: speakto

8.10.125. plugin_checkcommand

`plugin_checkcommand( sCommand[], &iAccess = 0 );`

`sCommand[]`
Typ: String (30)

`&iAccess = 0`
Typ: Integer (0 - 2147483647)

Die Funktion gibt zurück, wieviele Plugins den Admin Mod Befehl (sCommand) registriert haben. Darüber hinaus wird der dafür notwendige Access Level (iAccess) zurückgegeben. In Kombination mit plugin_exec lässt sich ein Plugin-Addon-System aufbauen.

Beispiel aus plugin_cw_creator[30] (Funktion: warmup_end):

```
    if (plugin_checkcommand("cwc_addon_start",iAccess)>0){
       plugin_exec("cwc_addon_start","");
       iAccess=0;
    }
```

In diesem Fall wird ausgewertet, ob mindestens ein Plugin den Admin Mod Befehl „cwc_addon_start“ registriert hat. Falls mindestens ein Plugin gefunden wurde, wird der Befehl ausgeführt.

Gehört zu: plugin.inc

[30] http://www.adminmod.de/plugins.php?plugin=plugin_cw_creator3

8.10.126. plugin_command

`plugin_command( HLCommand, HLData, HLUserName, UserIndex )`

`HLCommand`
Typ: HLString (33)

`HLData`
Typ: HLString (200)

`HLUserName`
Typ: HLString (30)

`UserIndex`
Typ: Integer (0 - 32)

Bei der Funktion plugin_command handelt es sich um ein Event, mit dem alle Befehle abgefangen werden können. Sie wird vor allem in Antiflood Plugins eingesetzt.

Beispiel aus plugin_blatt_map[31] (Funktion: plugin_command):

```
public plugin_command(HLCommand,HLData,HLUserName,UserIndex) {
  if (g_BuySupressed==0) {
    return PLUGIN_CONTINUE;
  }

  new Command[MAX_COMMAND_LENGTH];
  safe_convert(HLCommand,Command,MAX_COMMAND_LENGTH);

  if (strmatch(Command,"say",3)==1) {
    return PLUGIN_CONTINUE;
  }
  if (streq(Command,"drop")) {
    return PLUGIN_CONTINUE;
  }

  return PLUGIN_HANDLED;
}
```

In diesem Beispiel wird die Ausführung aller Befehle außer „say“ und „drop“ unterbunden (PLUGIN_HANDLED). Außerdem kann eine globale Variable (g_BuySupressed) das Unterbinden verhindern.

Dieser Code macht nur Sinn, wenn das Plugin als zweites (hinter plugin_antiflood ausgeführt wird. Nur Befehle aus Plugins, die sich in der Ladereihenfolge hinter plugin_blatt_map befinden, können blockiert werden.

Gehört zu: admin.inc

Siehe auch: plugin_connect, plugin_disconnect, plugin_info, plugin_init

[31] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_map

8.10.127. plugin_connect

`plugin_connect( HLUserName, HLIP, UserIndex )`

`HLUserName`
Typ: HLString (30)

`HLIP`
Typ: HLIP (22)

`UserIndex`
Typ: Integer (0 - 32)

Bei der Funktion plugin_connect handelt es sich um ein Event, mit dem man Informationen zum den Server beitretenden Spieler erhalten (Name, IP, Slot) kann. Leider ist der Event in der Regel zu früh, so dass weder Spielername noch seine IP zurückgegeben werden. Nur der Slot (UserIndex) lässt sich zuverlässig nutzen. Mehr Informationen sind dem Tutorial zu entnehmen.

Beispiel aus plugin_message (Funktion: plugin_connect):

```
public plugin_connect(HLUserName, HLIP, UserIndex) {
   set_timer("say_hello",45,0);
   return PLUGIN_CONTINUE;
}
```

Beim Beitreten eines Spielers wird ein 45 Sekunden Timer auf die Funktion „say_hello“ ausgelöst.

Gehört zu: admin.inc

Siehe auch: plugin_command, plugin_disconnect, plugin_info, plugin_init

8.10.128. plugin_disconnect

`plugin_disconnect( HLUserName, UserIndex )`

`HLUserName`
Typ: HLString (30)

`UserIndex`
Typ: Integer (0 - 32)

Bei der Funktion plugin_disconnect handelt es sich um ein Event, mit dem man Informationen zum den Server verlassenden Spieler erhalten (Name, Slot) kann.

Beispiel aus plugin_bk_hltvannounce[32] (Funktion: plugin_disconnect):

```
public plugin_disconnect(HLUserName, UserIndex){
   if (hltvid==UserIndex){
      hltvid=0;
      hltvteam=0;
#if LANG==1
      typesay("Der HLTV ist jetzt offline!",3,181,40,44);
#else
      typesay("The HLTV is offline!",3,181,40,44);
#endif
   }
   return PLUGIN_CONTINUE;
}
```

Falls es sich beim übergebenen Userindex (UserIndex) um den des HLTV-Servers (hltvid) handelt, wird Verlassen des HLTV-Servers mit typesay bekannt gegeben. Die Sprache wird beim Compilieren des Plugins durch die Direktive LANG eingestellt.

Gehört zu: admin.inc

Siehe auch: plugin_command, plugin_connect, plugin_info, plugin_init

[32] http://www.adminmod.de/plugins.php?plugin=plugin_bk_hltvannounce

8.10.129. plugin_exec

`plugin_exec( sCommand[], sArguments[] );`

`sCommand[]`
Typ: String (30)

`sArguments[]`
Typ: String (200)

Um zu verhindern, dass man mit der Funktion exec pluginübergreifend das Rechtesystem untergräbt, wurde plugin_exec eingeführt. Diese Funktion überprüft, ob der ausführende Spieler auch die Rechte für den Befehl des anderen Admin Mod Plugins hat.

Beispiel aus plugin_cw_creator[33] (Funktion: warmup_end):

```
    if (plugin_checkcommand("cwc_addon_start",iAccess)>0){
       plugin_exec("cwc_addon_start","");
       iAccess=0;
    }
```

In diesem Beispiel wird ausgewertet, ob mindestens ein Plugin den Admin Mod Befehl „cwc_addon_start“ registriert hat. Falls mindestens ein Plugin gefunden wurde, wird der Befehl mit plugin_exec ausgeführt. Der Befehl erwartet keine Argumente, so dass sie leer gelassen wurden.

Gehört zu: admin.inc

Siehe auch: exec

[33] http://www.adminmod.de/plugins.php?plugin=plugin_cw_creator3

8.10.130. plugin_info

`plugin_info( HLOldName, HLNewName, UserIndex )`

`HLOldName`
Typ: HLString (33)

`HLNewName`
Typ: HLString (33)

`UserIndex`
Typ: Integer (0 - 32)

Bei der Funktion plugin_info handelt es sich um ein Event, mit dem man den Namenswechsel eines Spielers auswerten kann. Das Event wird auch beim Connect mehrfach ausgeführt, so dass es hin und wieder als Ersatz für plugin_connect herangezogen wird. Da der Event nicht immer mit einem echten Namenswechsel verbunden ist, sollte man ihn mit Weitsicht verwenden.

Beispiel aus plugin_retribution (Funktion: plugin_info):

```
public plugin_info(HLOldName,HLNewName,UserIndex) {
   new iIgnoreGag = 0;
   new GagTime;
   new Command[MAX_TEXT_LENGTH];
   new NewName[MAX_NAME_LENGTH];
   new OldName[MAX_NAME_LENGTH];

   convert_string(HLNewName, NewName, MAX_NAME_LENGTH);
   convert_string(HLOldName, OldName, MAX_NAME_LENGTH);

   /* Only bother if the name has changed. */
   if(streq(OldName,NewName)==0) {
```

Der alte und der neue Spielername (HLNewName und HLOldName) werden in Small Strings konvertiert (convert_string). Die weitere Auswertung erfolt nur, wenn der alte nicht dem neuen Namen entspricht (streq).

Gehört zu: admin.inc

Siehe auch: plugin_command, plugin_connect, plugin_disconnect, plugin_init

8.10.131. plugin_init

`plugin_init( )`

Bei der Funktion plugin_init handelt es sich um ein Event, das nur beim Plugin Start ausgeführt. Es entspricht der main Funktion in der Sprache C. Jedes Plugin muss diese Funktion besitzen (siehe auch im Tutorial).

Beispiel aus plugin_antiflood (Funktion: plugin_init):

```
public plugin_init()
{
   plugin_registerinfo("Admin Anti-Flood Plugin", "Auto kicks flooders", VERSION);

   return PLUGIN_CONTINUE;
}
```

Dieses Beispiel zeigt die simpelste Form von plugin_init. Es wird lediglich das Plugin bei Admin Mod registriert (plugin_registerinfo). Es können außerdem Befehle registriert aber auch beliebiger Code ausgeführt werden. Die Funktion plugin_exec funktioniert allerdings zu diesem Zeitpunkt prinzipbedingt noch nicht.

Gehört zu: admin.inc

Siehe auch: plugin_command, plugin_connect, plugin_disconnect, plugin_info

8.10.132. plugin_message

`plugin_message( sMessage[] );`

`sMessage[]`
Typ: String (100)

Die Funktion schreibt eine Nachricht (sMessage) in Console des aufrufenden Spielers. Im Gegensatz zur Funktion selfmessage wird auch das Plugin genannt, das die Nachricht ausgibt.

Beispiel aus plugin_CS (Funktion: LoadDefaultRestrictions):

```
    if(get_vaultdata(strName,strKey,MAX_ITEMS+1)) {
        plugin_message("Map-specific saved weapon restrictions found.");
    } else if(get_vaultdata("WeaponRestrictions",strKey,MAX_ITEMS+1)) {
        plugin_message("Default saved weapon restrictions found.");
    } else {
        plugin_message("No saved weapon restrictions found.");
    }
```

Es wird nach mapspezifischen und allgemeinen Waffeneinschränkungen gesucht. Wenn der Eintrag (strName) in der vault.ini gefunden wurde, wird in der Console ausgegeben, dass mapspezifische Restriktionen geladen wurden. Beim Eintrag „WeaponRestrictions“ wird ausgegeben, dass die Standardeinschränkungen geladen wurden. Wurde kein Eintrag ausgemacht, erscheint in der Console, dass keine Restriktionen geladen wurden.

Gehört zu: plugin.inc

Siehe auch: selfmessage

8.10.133. plugin_registercmd

`plugin_registercmd( Command[], Function[], RequiredAccess, HelpMessage[]= "" );`

`Command[]`
Typ: String (30)

`Function[]`
Typ: String (19)

`RequiredAccess`
Typ: Integer (0 - 2147483647)

`HelpMessage[]= ""`
Typ: String (150)

Die Funktion dient der Registrierung eines Befehls (Command), der z.B. in der Console eingegeben wird. Als Argumente müssen die aufzurufende Funktion (Function) und der nötige Access Level (RequiredAccess) angegeben werden. Optional kann ein Hilfetext (z.B. die Syntax) angegeben werden, der dann beim Aufruf von admin_help ausgegeben wird. Man verwendet diesen Befehl üblicherweise nur in plugin_init. Mehr zu diesem Befehl ist auch dem Tutorial zu entnehmen.

Beispiel aus plugin_bk_alltalk[34] (Funktion: plugin_init):

```
plugin_registerinfo("Alltalk changer","Switches sv_alltalk",STRING_VERSION);
plugin_registercmd("admin_alltalk","admin_alltalk",A_KICK,"Switches sv_alltalk.");
```

Zunächst wird das Plugin registriert. Anschließend wird der Befehl „admin_alltalk“ mit der gleichnamigen Funktion registriert. Der Spieler benötigt Kickrechte. Hier wurde aus Darstellungsgründen „A_KICK“ geschrieben. Richtig ist jedoch „ACCESS_KICK“. Einen Überblick über die Access Level und die zugehörigen Direktiven ist der admin.inc zu entnehmen. Zu letzt wird ein Hilfetext für die Anzeige in admin_help festgelegt.

Gehört zu: admin.inc

Siehe auch: plugin_registerhelp, plugin_registerinfo

[34] http://www.adminmod.de/plugins.php?plugin=plugin_bk_alltalk

8.10.134. plugin_registerhelp

`plugin_registerhelp( Command[], RequiredAccess, HelpMessage[] );`

`Command[]`
Typ: String (30)

`RequiredAccess`
Typ: Integer (0 - 2147483647)

`HelpMessage[]`
Typ: String (150)

Einen Hilfetext kann man bereits mit plugin_registercmd anlegen. Es gibt jedoch auch Fälle, bei denen z.B. zwei Argumente übergeben werden müssen. Basierend auf dem ersten Argument müssen unterschiedliche zweite Argumente eingegeben werden. Man kann dann für alle ersten Argumente Hilfetexte anlegen. Wichtig ist die Funktion auch beim Abfangen von „say“ Nachrichten. Da viele Plugins den Chat für die Befehlseingabe nutzen, sollte man die eigenen Chatbefehle mittels plugin_registerhelp mit einem Hilfetext ausstatten. Die Registrierung von „say“ sollte hingegen keinen Hilfetext beinhalten.

Es muss der dem Hilfetext (HelpMessage) zugehörige Befehl (Command) und der notwendige Access Level (RequiredAccess) angegeben werden. Beim Aufruf von admin_help wird der Hilfetext dann angezeigt.

Beispiel aus plugin_bk_timeleft[35] (Funktion: plugin_init):

```
plugin_registerinfo("Akurates Timeleft","Genaue Restzeitangabe.",STRING_VERSION);
plugin_registercmd("say","HandleSay",ACCESS_ALL);
plugin_registerhelp("say",ACCESS_ALL,"say timeleft: Gibt die Restzeit an.");
```

Zunächst wird das Plugin registriert und anschließend der „say“ Befehl. D.h. alle Chatnachrichten (außer der Teamnachrichten) werden abgefangen. Da auch andere Plugins „say“ abfangen könnten, wird die Hilfe in plugin_registerhelp beschrieben.

Gehört zu: admin.inc

Siehe auch: plugin_registercmd

[35]http://www.adminmod.de/plugins.php?plugin=plugin_bk_timeleft

8.10.135. plugin_registerinfo

`plugin_registerinfo( Name[], Description[], Version[] );`

`Name[]`
Typ: String (100)

`Description[]`
Typ: String (200)

`Version[]`
Typ: String (150)

Dies Funktion registriert ein Plugin bei Admin Mod. Es muss ein Pluginname (Name), eine Beschreibung (Description) und eine Versionsnummer (Version) angegeben werden. Diese Informationen lassen sich mit admin_version abrufen.

Beispiel aus plugin_bk_alltalk[36] (Funktion: plugin_init):

```
plugin_registerinfo("Alltalk changer","Switches sv_alltalk",STRING_VERSION);
plugin_registercmd("admin_alltalk","admin_alltalk",A_KICK,"Switches sv_alltalk.");
```

Zunächst wird das Plugin registriert. Der Name ist „Alltalk changer“, die Beschreibung lautet „Switches sv_alltalk“ und die Versionsnummer entstammt einer globalen Variable (STRING_VERSION). Anschließend wird der Befehl „admin_alltalk“ registriert.

Gehört zu: admin.inc

[36] http://www.adminmod.de/plugins.php?plugin=plugin_bk_alltalk

8.10.136. pointto

`pointto( iRange = 2048 );`

`iRange = 2048`
Typ: Integer (0 - 2147483647)

Mit dieser Funktion kann man ermitteln, auf welchen Spieler der den Befehl aufrufende Spieler zielt. Die Funktion gibt die Session ID des anderen Spielers zurück. Die Session ID für eine ungültige Figur (z.B. Geisel) ist -1. Ist keine Figur im Blickfeld, wird eine 0 zurückgegeben. Das Argument iRange definiert die Blickweite.

Um auch den Blick anderer Spieler zu untersuchen, sollte man die Funktion gettarget nutzen. Zudem erspart man sich die Umwandlung der Session ID.

Beispiel aus plugin_seeuser[37] (Funktion: admin_seeuser):

```
        targetnumber = pointto();
        numtostr(targetnumber, numstring);
        get_username(numstring,targetname,MAX_NAME_LENGTH);
        get_userindex(targetname,targetnumber);
```

Zunächst wird die Session ID (targetnumber) des gegenüberstehenden Spielers ermittelt. Diese wird in einen String umgewandelt (numtostr), in einen Namen (targetname) und schließlich in einen Userindex (targetnumber) umgewandelt.

Gehört zu: admin.inc

Siehe auch: gettarget

8.10.137. print

`print( const string[], foreground=-1, background=-1 );`

`string[]`
Typ: String (0 - 2147483647)

`foreground=-1`
Typ: Integer (-1 - 7)

`background=-1`
Typ: String (-1 - 7)

Diese Funktion schreibt einen Text (string) in die Befehlszeile. Es können auch Vorder- und Hintergrundfarben (foreground und background) genutzt werden. Mangels einer solchen Befehlszeile ist die Funktion in Admin Mod nutzlos.

Gehört zu: console.inc

[37] http://www.adminmod.de/plugins.php?plugin=plugin_seeuser

8.10.138. printf

`printf( const format[], ... );`

`format[]`
Typ: String (0 - 2147483647)

Diese Funktion schreibt einen Text (format) in die Befehlszeile. Es können andere Strings unterschiedlicher Datentypen in format eingebettet werden. Mangels einer solchen Befehlszeile ist die Funktion in Admin Mod nutzlos.

Gehört zu: console.inc

8.10.139. rainbow

`rainbow( sText[], iTime, iRedStart, iGreenStart, iBlueStart, iRedEnd, iGreenEnd, iBlueEnd );`

`sText[]`
Typ: String (80)

`iTime`
Typ: Integer (0 - 2147483647)

`iRedStart`
Typ: Integer (0 - 255)

`iGreenStart`
Typ: Integer (0 - 255)

`iBlueStart`
Typ: Integer (0 - 255)

`iRedEnd`
Typ: Integer (0 - 255)

`iGreenEnd`
Typ: Integer (0 - 255)

`iBlueEnd`
Typ: Integer (0 - 255)

Mit dieser Funktion kann man eine Nachricht (sText) mit Farbverlauf in der Mitte des Bildschirmes produzieren. Das Argument iTime legt die Anzeigezeit in Sekunden fest. Die Start- und die Endfarbe werden in RGB vorgegeben (iRedStart, iGreenStart, iBlueStart bzw. iRedEnd, iGreenEnd, iBlueEnd).

Es ist kein Zeilenumbruch mit ^n möglich.

Beispiel aus plugin_budfroggy_rainbow[38] (Funktion: admin_rsay):

```
if(ic1==1 && ic2==1){
   rainbow(Text,10,iRedS,iGreenS,iBlueS,iRedE,iGreenE,iBlueE);
}else{
   iRedS = 255;
  iGreenS = 10;
  iBlueS = 10;
  iRedE = 10;
  iGreenE = 10;
  iBlueE = 255;
  rainbow( Data, 10, iRedS, iGreenS, iBlueS, iRedE, iGreenE, iBlueE);
}
```

Falls Farben definiert wurden (ic1 und ic2 müssen 1 sein), wird die Nachricht für zehn Sekunden mit diesem Farbverlauf dargestellt. Anderenfalls wird ein Verlauf von Rot nach Blau durchgeführt.

Gehört zu: admin.inc

Siehe auch: centersay, typesay

[38] http://www.adminmod.de/plugins.php?plugin=plugin_budfroggy_rainbow

8.10.140. random

`random( max );`

`max`
Typ: Integer (1 bis 2147483647)

Die Funktion gibt eine Pseudo-Zufallszahl zurück. Der Bereich der Zufallszahlen ist zum einen durch 0 und zum anderen durch max - 1 begrenzt. Man darf von dieser Funktion keine Zufallszahlen erwarten, die wissenschaftlichen Ansprüchen genügen. Aber dafür ist sie bei Admin Mod auch nicht gedacht.

Beispiel aus plugin_fun (Funktion: DiscoInferno):

```
      for (i = 1; i <= iMaxPlayers; i++) {
         if(playerinfo(i,Name,MAX_NAME_LENGTH)==1) {
            Red = random(256);
            Green = random(256);
            Blue = random(256);
            glow(Name,Red,Green,Blue);
         }
      }
```

Mittels For-Schleife und der Funktion playerinfo werden alle Serverslots auf existierende Spieler überprüft. Ist ein Spieler gefunden worden, werden für jeden Farbanteil eine Zufallszahl für den Bereich 0 bis 255 erzeugt. Der Spieler glüht dann in der Farbe, die sich aus den Zufallszahlen ergab (glow).

Gehört zu: core.inc

8.10.141. readfile

`readfile( sFilename[], sLine[], iLineNum, iMaxLength );`

`sFilename[]`
Typ: String (200)

`sLine[]`
Typ: String (200)

`iLineNum`
Typ: Integer (0 - 2147483647)

`iMaxLength`
Typ: Integer (0 - 200)

Mit dieser Funktion kann eine Zeile (iLineNum) aus einer Datei (sFilename) ausgelesen werden. Die Zeile darf nicht länger als 200 Zeichen sein (iMaxLength). Die Rückgabe wird in der Variable sLine gespeichert.

Die Funktion ist sehr zeitintensiv, da gewartet werden muss bis der Festplattenzugriff erfolgt ist. Lädt man mehrere Zeilen mit einer For-Schleife, wird bei jeder Zeile die Datei neu geöffnet, was weitere Verzögerungen verursacht. Man sollte sich daher genau überlegen, ob man diese Funktion nutzen möchte. Beim Auslesen während des Serverstarts ist dies noch akzeptabel, ein Auslesen während der Spielzeit kann bei einer For-Schleife durchaus zu Aussetzern im Spiel führen (Lag). Wenn möglich sollte man seine Einstellungen über die vault.ini lesen und schreiben.

Beispiel aus plugin_bk_res[39] (Funktion: add_res_data):

```
        iLines=filesize(sMap,lines);
        for(i=1;i<=iLines;i++){
           readfile(sMap,sLine,i,MAX_DATA_LENGTH);
           writefile(sMap2,sLine,-1);
        }
```

Es wird die Anzahl der Zeilen der Datei (sMap) ausgelesen. Anschließend werden die Daten über eine For-Schleife zeilenweise von einer Datei (sMap) in eine andere (sMap2) kopiert. Die Zeilen übernehmen nicht unbedingt die Zeilennummer der Ursprungsdatei. Die Zeilen werden beim Schreiben nur angehängt (-1).

Gehört zu: admin.inc

Siehe auch: deletefile, filesize, resetfile, writefile

[39] http://www.adminmod.de/plugins.php?plugin=plugin_bk_res

8.10.142. reject_message

`reject_message( iPublic = 0 );`

`iPublic = 0`
Typ: Integer (0 - 1)

Die Funktion gibt eine Fehlermeldung an den Spieler aus „You do not have access to this command“, wenn ein Spieler ohne die notwendigen Rechte versucht einen Befehlauszuführen. Diese Funktion greift auf die in admin_reject_msg definierte Nachricht zurück. Ist diese dort nicht definiert, wird dem Spieler „You do not have access to this command“ in der Konsole angezeigt. Verwendet man iPublic = 1, wird die Nachricht serverweit im Chat ausgegeben.

Beispiel aus plugin_base (Funktion: admin_rcon):

```
   if (strstr(Data, "rcon_password") >= 0) {
      reject_message();
      return PLUGIN_HANDLED;
   }
```

Hierbei handelt es sich um eine Absicherung, dass kein Admin, der Zugriff zu admin_rcon hat, sich unberechtigt Zugriff zu RCon verschafft, indem er das Passwort ändert. Falls der String Data die Servervariable rcon_password beinhaltet, wird die Meldung „You do not have access to this command“ angezeigt und die weitere Abarbeitung abgebrochen.

Gehört zu: adminlib.inc

8.10.143. reload

`reload( );`

Die Funktion lädt alle Ini-Dateien von Admin Mod neu, wenn diese in der adminmod.cfg definiert wurden.

- admin_vault_file
- ips_file
- models_file
- users_file
- words_file

Dies kann sinnvoll sein, wenn man mit dem Plugin Änderungen an den Dateien vorgenommen hat und diese aktivieren möchte.

Beispiel aus plugin_base (Funktion: admin_reload):

```
public admin_reload(HLCommand,HLData,HLUserName,UserIndex) {
   reload();
   return PLUGIN_HANDLED;
}
```

Hier werden auf Befehl einfach die Dateien neu geladen. Dies ist z.B. interessant, wenn man Änderungen an der users_ini vorgenommen hat und diese ohne Serverneustart oder Mapchange übernehmen möchte.

Gehört zu: admin.inc

8.10.144. removespawn

`removespawn( iIdentity );`

`iIdentity`
Typ: Integer (0 - 2147483647)

Die Funktion löscht ein Item (iIdentity).

Die Funktionalität steht mittlerweile nicht mehr zur Verfügung.

Beispiel aus plugin_spawn (Funktion: admin_removespawn):

```
    if(removespawn(iIdentity)==1) {
       say_command(User,Command,Data);
       selfmessage( "Success.");
    } else {
       selfmessage( "Failed.");
    }
```

Es wird versucht ein Item zu löschen. Falls dies erfolgreich war, wird eine Erfolgsmeldung an den Admin und alle anderen Spieler ausgegeben. Anderenfalls wird nur der Admin über den fehlgeschlagenen Versuch informiert.

Gehört zu: admin.inc

Siehe auch: listspawn, movespawn, spawn

8.10.145. resetfile

`resetfile( sFilename[] );`

`sFilename[]`
Typ: String (100)

Unter Angabe des Dateinamens löscht die Funktion den Inhalt komplett. Die Datei bleibt jedoch im Gegensatz zu deletefile erhalten.

Beispiel aus plugin_cw_creator3[40] (Funktion: create_mapcycle):

```
    resetfile(sPath);
    get_vaultdata("CWC_MAP1",Text,MAX_DATA_LENGTH);
    writefile(sPath,Text,-1);
```

Die Datei (sPath) wird geleert. Aus der vault.ini wird ein Mapname in Text eingelesen und anschließend in die leere Datei geschrieben.

Gehört zu: admin.inc

Siehe auch: deletefile, filesize, readfile, writefile

[40] http://www.adminmod.de/plugins.php?plugin=plugin_cw_creator3

8.10.146. rindex

`rindex( sSource[], iChar );`

`sSource[]`
Typ: String (200)

`iChar`
Typ: Integer (0 - 256)

Die Funktion ermittelt die Position des ersten Vorkommens des Zeichens (iChar) im String (sSource) von rechts beginnend. Für das Zeichen muss die ASCII-Nummer angegeben werden. Vereinfacht kann auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden. Die Funktion liefert -1 zurück, falls das Zeichen nicht gefunden wurde.

Die Funktion ist ein Alias für strrchr.

Beispiel:

```
new sTest[MAX_TEXT_LENGTH];
strcpy(sTest,"Dies ist ein Test.",MAX_TEXT_LENGTH);
new iTest=0;
iTest=rindex(sTest,' ');
```

Es wird zunächst „Dies ist ein Test.“ in die Variable sTest geschrieben (strcpy). Nun wird von rechts nach dem ersten Auftreten eines Leerzeichens gesucht. Leerzeichen gibt es an den Positionen 4, 8 und 12. Dem entsprechend wird an iTest 12 übergeben.

Gehört zu: string.inc

Siehe auch: index, strchr, strrchr

8.10.147. say

```
say( sMessage[] );
```

`sMessage[]`
Typ: String (100)

Die Funktion gibt einen vorgegebenen Text (sMessage) im Chat aus.

Beispiel aus plugin_CS (Funktion: HandleKickVote):

```
648         new Ratio = getvar("kick_ratio");
649         if (VoteCount >= Ratio*UserCount/100) {
650            if (g_AbortVote) {
651               say("Kick vote was aborted by an admin");
652            } else {
653               set_timer("KickUser",10,1,VoteUser);
654            }
```

Die Servervariable „kick_ratio“ wird als Ganzzahl in Ratio ausgelesen. Falls genügend Stimmen für den Kick zusammengekommen sind, wird entweder der Kick abgebrochen, weil ein Admin ihn aufgehalten hat, oder nach 10 Sekunden der entsprechende Spieler vom Server geworfen. Falls ein Admin den Vote aufgehalten hat, wird dies allen Spielern im Chat mitgeteilt (Zeile 651).

Gehört zu: admin.inc

Siehe auch: centersay, typesay, messageex

8.10.148. say_command

`say_command( sUser[], sCommand[], sData[], iOverride = 0 );`

`sUser[]`
Typ: String (33)

`sCommand[]`
Typ: String (19)

`sData[]`
Typ: String (200)

`iOverride = 0`
Typ: Integer (0 - 1)

Die Funktion schreibt einen formatierten Text in den Chat. Dazu muss der ausführende Admin (sUser), der ausgeführte Befehl (sCommand) und die Parameter (sData) angegeben werden. Die Ausgabe kann durch die Einstellung von admin_quiet beeinflusst werden. Optional kann diese Einstellung auch ignoriert werden (iOverride = 1).

Beispiel aus plugin_base (Funktion: admin_map):

```
    if (valid_map(Data)==1) {
       say_command(User,Command,Data);
       changelevel(Data, 4);
```

Wenn es sich bei Data um eine gültige Map handelt, wird basierend auf der Einstellung von admin_quiet im Chat mit Adminnamen und Befehl ausgegeben und nach 4 Sekunden ein Mapwechsel durchgeführt.

Gehört zu: adminlib.inc

Siehe auch: format_command, log_command

8.10.149. selfmessage

`selfmessage( sMessage[] );`

`sMessage[]`
Typ: String (100)

Die Funktion schreibt dem Spieler, der den zugehörigen Befehl ausgeführt hat, eine Nachricht (sMessage) in die Console.

Beispiel aus plugin_base (Funktion: admin_rcon):

```
    if (check_auth(ACCESS_RCON)==0) {
       selfmessage("Laf. Silly bear.");
       return PLUGIN_HANDLED;
    }
```

Um zu verhindern, dass jemand auf die Idee kommt aus der Serverconsole den Befehl admin_rcon aufzurufen, was möglich aber unsinnig ist, wurde eine Abfangroutine implementiert. Nur wenn der User den Access Level 65536 (ACCESS_RCON) besitzt, wird der RCon-Befehl über Admin Mod abgesetzt. Anderenfalls wird der Admin, der den Befehl ausführte, etwas unhöflich in der Console auf das unsinnige Ansinnen hingewiesen (selfmessage).

Gehört zu: admin.inc

Siehe auch: directmessage, message, messageex

8.10.150. servertime

`servertime( sTimeString[], iMaxLen, sFormat[] = „none" );`

`sTimeString[]`
Typ: String (100)

`iMaxLen[]`
Typ: String (1 - 100)

`sFormat[]`
Typ: String (100)

Die Funktion gibt das aktuelle Datum und die Uhrzeit zurück (sTimeString mit Länge iMaxLen). Die Rückgabe erfolgt über die Formatvorgabe (sFormat). Falls diese nicht vorgegeben wurde, werden die abgelaufenen Sekunden seit dem 1.1.1970 ausgegeben.

In der folgenden Liste sind die Formatierungszeichen beschrieben:

%a Wochentag (abgekürzt) (z.B. Mon für Montag)

%A Wochentag (ausgeschrieben)

%b Monat (abgekürzt) (z.B. Aug für August)

%B Monat (ausgeschrieben)

%c Vorgefertigtes Datums- und Zeitformat (%a %b %d %H:%M:%M %Y)

%d Tag im Monat (01 - 31)

%H Stunde im 24h Format (00 - 23)

%I Stunde im 12h Format (00 - 12)

%j Tag im Jahr (001 - 366)

%m Monat (als Zahl) (01 - 12)

%M Minute (00 - 59)

%p Vor- und Nachmittagsperiode (AM oder PM)

%S Sekunden (00 - 61)

%U Kalenderwoche (Woche beginnt am ersten Sonntag) (00 - 53)

%w Wochentag (als Zahl, Sonntag = 0) (0 - 6)

%W Kalenderwoche (Woche beginnt am ersten Montag) (00 - 53)

%x Vorgefertigtes Datumsformat (%m/%d/%y)

%X Vorgefertigtes Zeitformat (%H:%M:%M)

%y Jahr (letzten zwei Ziffern) (z.B. 09)

%Y Jahr (alle Ziffern) (z.B. 2009)

%Z Zeitzone (z.B. CET)

Beispiel aus plugin_dotw[41] (Funktion: admin_time):

```
servertime(t_time, MAX_DATA_LENGTH, "%I:%M %p");
servertime(t_date, MAX_DATA_LENGTH, "%A, %B %d %Y");

snprintf(Text, MAX_TEXT_LENGTH, "%s %s", t_time, t_date);
selfmessage(Text);
```

Es wird die Serverzeit (t_time) und das Datum (t_date) ermittelt. Anschließend werden die beiden Zeitstrings mit einem Leerzeichen verknüpft und an den admin_time ausführenden Spieler in der Console ausgegeben. Dieser Code hätte auch mit nur einer servertime Abfrage und ohne snprintf geschrieben werden können.

Gehört zu: admin.inc

Siehe auch: systemtime

[41] http://www.adminmod.de/plugins.php?plugin=plugin_dotw

8.10.151. set_serverinfo

`set_serverinfo( sKey[], sValue[] );`

`sKey[]`
Typ: String (100)

`sValue[]`
Typ: String (200)

Die Funktion schreibt die Daten (sValue) mit dem Schlüssel (sKey) in ein kleinen Speicherbereich. Die Daten werden beim Mapwechsel nicht gelöscht. Der Speicher ist jedoch nur sehr gering bemessen, so dass die Verwendung dieser Funktion vermieden werden sollte.

Beispiel aus plugin_bk_cron[42] (Funktion: lag_check):

```
numtostr(iTime,sTime);
set_serverinfo("last_cron",sTime);
```

Die Sekundenzahl der aktuellen Minute (iTime) wird in einen String (sTime) umgewandelt. Anschließend wird ein Schlüssel „last_cron" erstellt oder mit diesem String überschrieben. Auf diese Art und Weise kann insbesondere überprüft werden, wie lang der letzte Durchlauf vor dem Mapwechsel her ist. Beim Serverstart ist die Variable noch nicht gesetzt, so dass 0 zurückgegeben wird. Die Nutzung der vault.ini verbietet sich, da der Wert beim Beenden bzw. Absturz des Servers nicht automatisch auf 0 zurückgesetzt wird.

Gehört zu: admin.inc

Siehe auch: get_serverinfo, get_userinfo

[42] http://www.adminmod.de/plugins.php?plugin=plugin_bk_cron

8.10.152. set_timer

`set_timer( sFunction[], iWaitSeconds, iRepeatCount, sParameter[]= "" );`

`sFunction[]`
Typ: String (19)

`iWaitSeconds`
Typ: Integer (0 - 2147483647)

`iRepeatCount`
Typ: Integer (0 - 2147483647)

`sParameter[]`
Typ: String (200)

Mit dieser Funktion wird ein Timer erstellt. Dabei muss die bei Ablauf des Timers aufzurufende Funktion (sFunction), die Ablaufzeit (iWaitSeconds) und die Anzahl der Wiederholungen (iRepeatCount) übergeben werden. Die definierte Funktion muss öffentlich (public) sein. Unendlich viele Wiederholungen erreicht man durch die Angabe von 99999. Um nur einen einzigen Ablauf zu erreichen kann iRepeatCount auf 0 oder 1 gesetzt werden. Optional kann ein String (sParameter) an die aufzurufende Funktion übergeben werden. Die Funktion set_timer gibt die erstellte Timer ID direkt zurück. Mehr Informationen zur Verwendung von Timern können dem Tutorial entnommen werden.

Beispiel aus plugin_CS (Funktion: HandleKickVote):

```
        new Ratio = getvar("kick_ratio");
        if (VoteCount >= Ratio*UserCount/100) {
           if (g_AbortVote) {
              say("Kick vote was aborted by an admin");
           } else {
              set_timer("KickUser",10,1,VoteUser);
           }
```

Die Servervariable „kick_ratio“ wird als Ganzzahl in Ratio ausgelesen. Falls genügend Stimmen für den Kick zusammengekommen sind, wird entweder der Kick abgebrochen, weil ein Admin ihn aufgehalten hat, oder ein Timer ausgelöst, der einmal nach 10 Sekunden die Funktion „KickUser“ ausführt. Der entsprechende Spieler wird von dieser Funktion dann vom Server geworfen.

Gehört zu: admin.inc

Siehe auch: get_timer, kill_timer

8.10.153. set_vaultdata

`set_vaultdata( sKey[], sData[] );`

`sKey[]`
Typ: String (100)

`sData[]`
Typ: String (200)

Die Funktion schreibtDaten (sData) mit dem Schlüssel (sKey) in die vault.ini. Die Daten bleiben über einen Mapchange bzw. einen Serverneustart erhalten. Das Schlüssel-Daten-Paar kann mit set_vaultdata oder get_vaultnumdata ausgelesen werden. Die Funktion ist ideal um pluginspezifische Einstellungen zu speichern.

Weiß man bereits, dass die zu schreibenden Daten als Ganzzahl vorliegen, sollte man der Funktion set_vaultnumdata den Vorzug geben.

Mit set_vaultdata kann auch ein Schlüssel gelöscht werden. Dazu muss ein leerer Datenstring übergeben werden.

Beispiel aus plugin_retribution (Funktion: AddUserFlag):

```
if(get_vaultdata(sAuthID,VaultData,MAX_DATA_LENGTH) != 0) {
   if (strcasestr(VaultData," llama") == -1) {
      strcat(VaultData," llama", MAX_DATA_LENGTH);
      set_vaultdata(sAuthID,VaultData);
   }
```

Es wird versucht den Schlüssel mit der Steam ID des Spielers zu finden. Ist dies der Fall, wird das Ergebnis in VaultData zwischengespeichert. Falls das Ergebnis nicht bereits den Teilstring „ llama“ beinhaltet, wird dies dem Ergebis angehängt (strcat) und zurück in die vault.ini geschrieben.

Gehört zu: admin.inc

Siehe auch: get_vaultdata, get_vaultnumdata, set_vaultnumdata

8.10.154. set_vaultnumdata

`set_vaultnumdata( sKey[], iData );`

`sKey[]`
Typ: String (100)

`iData`
Typ: Integer (0 - 2147483647)

Die Funktion schreibt Daten (sData) mit dem Schlüssel (sKey) als Ganzzahl (iData) in die vault.ini. Die Daten bleiben über einen Mapchange bzw. einen Serverneustart erhalten. Das Schlüssel-Daten-Paar wird mit get_vaultdata oder get_vaultnumdata ausgelesen. Die Funktion ist ideal um pluginspezifische Einstellungen zu speichern.

Ist im Schlüssel ein String hinterlegt, darf man die Funktion nicht anwenden. Statt dessen ist set_vaultdata zu verwenden.

Beispiel aus plugin_bk_botmanager[43] (Funktion: plugin_init):

```
    if(!get_vaultnumdata("BK_BM_BOTS",iBots)){
       set_vaultnumdata("BK_BM_BOTS",0);
       if(getvar("admin_debug")>0){
          log("[BK_BM] Basic setting missing in vault.ini. Creating new.");
       }
    }
```

Es wird überprüft, ob der Schlüssel BK_BM_BOTS existiert. Wenn dies nicht der Fall ist, wird der Schlüssel angelegt und eine 0 hineingeschrieben. Falls admin_debug größer als 0 ist, wird dies auch in den Logdateien dokumentiert.

Gehört zu: admin.inc

Siehe auch: get_vaultdata, get_vaultnumdata, set_vaultdata

[43] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.155. setarg

`setarg( arg, index=0, value );`

`arg`
Typ: Integer (0 - 2147483647)

`index=0`
Typ: Integer (0 - 2147483647)

`value`
Typ: Integer (-2147483648 - 2147483647)

Diese Funktion wird nur Fällen gebraucht, wenn man die Argumente einer Funktion mit variabler Argumentanzahl verändern möchte. Dabei gibt „arg“ die Position des Arguments beginnend mit 0 an. Das Argument „index“ wird nur benötigt, wenn arg ein Feld (z.B. ein String) ist. Es gibt die zu schreibende Zelle des Feldes an. Das Argument „value“ ist der neue Wert, der zurückzuschreiben ist.

Beispiel:

```
sum(...){
   new result = 0 ;
   for (new i = 0; i < numargs(); ++i) {
      result += getarg(i);
   }
   setarg(0,result);
   return 1;
}
```

Eine Funktion „sum“ mit variabler Argumentanzahl (...) wird definiert. Eine For-Schleife fragt die einzelnen Argumente ab und addiert sie. Die Summe wird über das erste Argument der Funktion „sum“ zurückgegeben. Die direkte Rückgabe ist immer 1. Die Anzahl der übergebenen Argumente wird mit numargs ermittelt.

Gehört zu: core.inc

Siehe auch: getarg, numargs

8.10.156. setproperty

`setproperty( id=0, const name[]="", value=cellmin, const string[]="" );`

`id=0`
Typ: Integer (-2147483648 - 2147483647)

`const name[]=""`
Typ: String (variabel)

`value=cellmin`
Typ: Integer (-2147483648 - 2147483647)

`const string[]=""`
Typ: String (variabel)

Mit dieser Funktion kann eine so genannte Property bei gegebenem Schlüssel (name oder value) erstellt werden. Die Property wird als String im Speicher abgelegt. Es handelt sich um eine Funktion, deren Benutzung unter Admin Mod vermieden werden sollte. Stattdessen wird empfohlen z.B. die Funktionen get_vaultdata oder set_vaultdata zurückzugreifen.

Mehr Informationen zum Thema sind im Abschnitt 8.14 nachzulesen.

Beispiel:

```
setproperty(2,"test_prop",sString);
setproperty(3,15,sString);
```

Das erste Beispiel erstellt unter der ID 2 die Property „test_prop“ mit dem Inhalt aus der Variable sString, während das zweite Beispiel unter der ID 3 die Property 15 mit dem gleichen Inhalt erstellt.

Gehört zu: core.inc

Siehe auch: existproperty, deleteproperty, getproperty

8.10.157. setstrvar

`setstrvar( cvar[], value[] );`

`cvar[]`
Typ: String (100)

`value[]`
Typ: String (100)

Mit dieser Funktion können Servervariablen (cvar) gesetzt werden. Der Inhalt (value) muss als String vorliegen. Eine äquivalente Funktion für Ganzzahlen existiert nicht, so dass diese vorab in einen String umgewandelt werden müssen (numtostr).

Beispiel aus plugin_CS (Funktion: HandleRestartVote):

```
    new Timelimit[MAX_NUMBER_LENGTH];
    numtostr((getvar("mp_timelimit") * 60 - timeleft() + 10) / 60,Timelimit);
    setstrvar("mp_timelimit",Timelimit);
    setstrvar("sv_restartround","10");
    say("Round restart vote succeeded.");
```

Vom aktuellen Zeitlimit auf dem Server wird die Restzeit abgezogen und 10 Sekunden addiert. Dieser Wert wird in einen String umgewandelt (numtostr) und in die Servervariable „mp_timelimit“ als neues Zeitlimit eingetragen. Anschließend wird mit dem Setzen der Variable „sv_restartround“ auf 10 ein Maprestart nach 10 Sekunden angefordert und der Restart allen Spielern mitgeteilt.

Gehört zu: admin.inc

Siehe auch: exec, getstrvar, getvar

8.10.158. slap

`slap( sPlayer[] );`

`sPlayer[]`
Typ: String (33)

Die Funktion schlägt den angegebenen Spieler (sPlayer). Er wird dabei leicht in eine zufällige Richtung gestoßen, und es werden ihm fünf Lebenspunkte abgezogen. Ein Spieler kann durch diese Aktion nie weniger als einen Lebenspunkt bekommen. Der Schlag wird bei einem Lebenspunkt als abzugsfreier Stoß durchgeführt.

Beispiel aus plugin_retribution (Funktion: admin_slap):

```
if(check_immunity(Data)!=0) {
   snprintf(Text, MAX_TEXT_LENGTH, "Laf. You can't slap %s", TargetName);
   messageex(User, Text, print_chat);
} else {
   slap(Data);
}
```

Hat die Person (Data) Immunitätsrechte, wird eine Fehlermeldung an den Admin ausgegeben. Anderenfalls wird die Person geschlagen.

Gehört zu: admin.inc

Siehe auch: slay, teleport

8.10.159. slay

`slay( sPlayer[] );`

`sPlayer[]`
Typ: String (33)

Der Spieler (sPlayer) wird durch diese Funktion getötet. Wenn admin_fx aktiviert ist, wird die Aktion durch zusätzliche Effekte untermalt.

Beispiel aus plugin_retribution (Funktion: admin_slap):

```
if(check_immunity(TargetName)!=0) {
   snprintf(Text, MAX_TEXT_LENGTH, "Laf. You can't slay %s", TargetName);
   messageex(User, Text, print_chat);
} else {
   if ( slay(TargetName) ) {
      PlaySoundToAll("ambience/thunder_clap.wav");
```

Hat die Person (Data) Immunitätsrechte, wird eine Fehlermeldung an den Admin ausgegeben. Anderenfalls wird die Person getötet. Wenn die Aktion erfolgreich war, wird darüber hinaus über eine pluginspezifische Funktion (PlaySoundToAll) eine Sounddatei bei allen Spielern abgespielt.

Gehört zu: admin.inc

Siehe auch: slap, teleport

8.10.160. snprintf

`snprintf( sDest[], iLength, sFormat[], ... );`

`sDest[]`
Typ: String (200)

`iLength`
Typ: Integer (0 - 200)

`sFormat[]`
Typ: String (200)

`...`
variable Anzahl an Argumenten (kommagetrennt)

Mit dieser Funktion können auf einfache Weise Variablen unterschiedlicher Datentypen (...) in einen String (sDest mit Länge iLength) eingebettet werden. Über einen weiteren String sFormat wird festgelegt, wie die Variablen eingebunden werden sollen. Die Reihenfolge der Platzhalter im Format-String muss der Reihenfolger der Argumente im Anschluss an selbigen entsprechen.

Folgende Platzhalter stehen zur Verfügung:

%c Einzelnes Zeichen

%d Ganzzahl (man sieht oft %i, was aber nicht korrekt ist, aber trotzdem funktioniert)

%q Festkommazahl

%s Zeichenkette (String)

Beispiel aus plugin_blatt_map[44] (Funktion: DebugHeap):

```
DebugHeap(context[]) {
  if (g_DebugLevel >= 2) {
    new heapsize[MAX_TEXT_LENGTH];
    snprintf(heapsize,MAX_TEXT_LENGTH,"[%s] %i free bytes.",context,heapspace());
    plugin_message(heapsize);
  }
  return 0;
}
```

Wenn der in der globalen Variable (g_DebugLevel) Debuglevel größer oder gleich zwei ist, wird der noch verbleibende Heap-Speicher ermittelt (Ganzzahl). Mit dem String (context) soll diese Ganzzahl in der Variable heapsize eingebettet werden. Die Reihenfolge der Platzhalter (%s vor %i) bestimmt die Reihenfolge der Argumente (context vor

[44] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_map

heapspace()). Ist context gleich „Test“ und die Rückgabe von heapspace gleich 936, dann sähe der String heapsize folgendermaßen aus:

```
[Test] 936 free bytes.
```

Zuletzt wird dieser Text in der Console unter Angabe des Pluginnamens ausgegeben (plugin_message).

Gehört zu: string.inc

Siehe auch: strsubst

8.10.161. spawn

`spawn( sClass[], iX, iY, iZ, iXAngle, iYAngle, iZAngle );`

`sClass[]`
Typ: String (100)

`iX`
Typ: Integer (-2147483648 - 2147483647)

`iY`
Typ: Integer (-2147483648 - 2147483647)

`iZ`
Typ: Integer (-2147483648 - 2147483647)

`iXAngle`
Typ: Integer (0 - 359)

`iYAngle`
Typ: Integer (0 - 359)

`iZAngle`
Typ: Integer (0 - 359)

Die Funktion erstellt ein Item (iClass) auf einer bestimmten Position (iX, iY, iZ) und in einer bestimmten Ausrichtung (iXAngle, iYAngle, iZAngle). Es wird die ID des Items zurückgegeben.

Die Funktionalität steht mittlerweile nicht mehr zur Verfügung.

Beispiel aus plugin_spawn (Funktion: admin_spawn):

```
iIdentity = spawn(strClass,X,Y,Z,XAngle,YAngle,ZAngle);
if (iIdentity != 0) {
   snprintf(Text,MAX_TEXT_LENGTH,"Spawn created with ID %i.",iIdentity);
   selfmessage(Text);
   say_command(User,Command,Data);
} else {
   selfmessage("Failed.");
}
```

Es wird versucht ein Item an den genannten Koordinaten und in den gegebenen Winkeln zu erstellen. Falls dies erfolgreich war, wird eine Erfolgsmeldung an den Admin und alle anderen Spieler ausgegeben. Anderenfalls wird nur der Admin über den fehlgeschlagenen Versuch informiert.

Gehört zu: admin.inc

Siehe auch: listspawn, removespawn, spawn

8.10.162. speakto

`speakto( sTarget[], sSentence[] );`

`sTarget[]`
Typ: String (33)

`sSentence[]`
Typ: String (100)

Mit dieser Funktion kann man bei einem Spieler (sTarget) die in Half-Life integrierte Sprachausgabe nutzen. Dafür muss der Soundtyp und der zu sprechende Satz definiert werden (sSentence). Ein Überblick über die zur Verfügung stehenden Soundtypen und der jeweiligen Sounds kann dem Anhang entnommen werden.

Beispiel aus plugin_sdal_time_manager[45] (Funktion: speak_timeleft):

```
numtoword(minutes,Min,MAX_NUMBER_LENGTH);
numtoword(seconds,Sec,MAX_NUMBER_LENGTH);
snprintf(Text,MAX_DATA_LENGTH,"fvox/%s minutes %s seconds remaining",Min,Sec);
speakto(User,Text);
```

Mit der userdefinierten Funktion numtoword werden die Minuten und Sekunden von Ganzzahlen in ausgeschriebene, englischsprachige Strings umgewandelt (z.B. 10 in „ten“ oder 3 in „three“). Diese können von den Soundtypen vox (männliche Stimme) und fvox (weibliche Stimme) wiedergegeben werden. Da hier die weibliche Stimme benötigt wird, stellt man „fvox/“ dem Satz voran. Durch das Einbetten der Minuten und Sekunden kann der Satz (Text) beim Spieler (User) ausgegeben werden. Es können nur Wörter verwendet werden, die für den jeweiligen Soundtyp vorhanden sind.

Gehört zu: admin.inc

Siehe auch: playsound

[45] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_time_manager

8.10.163. strbreak

`strbreak( str[], first[], second[], maxlen, flen=sizeof first, slen=sizeof second );`

`str[]`
Typ: String (200)

`first[]`
Typ: String (200)

`second[]`
Typ: String (200)

`maxlen` Typ: Integer (1 - 200)

`flen=sizeof first` Typ: Integer (1 - 200)

`slen=sizeof second` Typ: Integer (1 - 200)

Diese Funktion teilt einen gegebenen String (str mit der Länge maxlen) beim ersten Leerzeichen in zwei Teile (first und second). Die Längen der zwei Teile sind optional, und im Allgemeinen nicht notwendig zu definieren. Es gibt allerdings mittlerweile bessere Funktionen (strsep und strsplit), die das Gleiche und noch mehr können.

Beispiel aus plugin_base (Funktion: admin_ban):

```
    convert_string(HLCommand,Command,MAX_COMMAND_LENGTH);
    convert_string(HLData,Data,MAX_DATA_LENGTH);
    convert_string(HLUserName,User,MAX_NAME_LENGTH);

    strbreak(Data,ban_user,strTime, MAX_DATA_LENGTH);
    strbreak(strTime, strTime, strType, MAX_DATA_LENGTH);
```

Zunächst werden die HL Strings in Small Strings konvertiert (convert_string). Anschließend wird der zu bannende Spieler (ban_user) von der Dauer des Banns (strTime) getrennt. Ein weiterer Trennversuch sorgt dafür, dass der gegebenenfalls in strTime vorhandene Typ des Banns (ID oder IP) in strType abgetrennt wird. Ist er nicht angegeben, ist der String leer.

Gehört zu: adminlib.inc

Siehe auch: strsep, strsplit

8.10.164. strcasecmp

`strcasecmp( sString1[], sString2[] );`

`sString1[]`
Typ: String (200)

`sString2[]`
Typ: String (200)

Die Funktion vergleicht zwei Strings (sString1 und sString2) über ihre gesamte Länge. Groß- und Kleinschreibung wird nicht beachtet. Sind beide Strings identisch, wird eine 0 zurückgegeben. Man erhält eine 0 zurück, wenn beide Strings gleich sind. Ist das erste ungleiche Zeichen in den Strings bei sString1 größer als bei sString2, wird eine Zahl größer als 0 zurückgegeben. Umgekehrt wird eine Zahl kleiner als 0 zurückgegeben.

Beispiel aus plugin_bk_botmanager[46] (Funktion: admin_bot_set):

```
	strbreak(sValue,sBotCvar,sValue,MAX_DATA_LENGTH);
	strtrim(sValue," ",2);
	strtrim(sBotCvar," ",2);

	if(strcasecmp(sBotCvar,"bots")==0){
```

Der Wert sValue wird in sBotCvar und sValue aufgeteilt (strbreak). Beiden Variablen werden am Anfang und am Ende von möglichen Leerzeichen befreit (strtrim). Die Ausführung der If-Verzweigung erfolgt nur, wenn die Variable sBotCvar genau dem String „bots“ entspricht, wobei die Groß- und Kleinschreibung keine Rolle spielt.

Gehört zu: string.inc

Siehe auch: strcmp, strncasecmp, strncmp

[46] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.165. strcasestr

`strcasestr( sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht einen String (sSearch) in einem anderen String (sSource). Groß- und Kleinschreibung wird nicht beachtet. Wird der Suchstring gefunden, gibt die Funktion eine Zahl größer als 0 zurück. Wenn der String nicht gefunden wurde, gibt die Funktion eine -1 zurück. Ist der Suchstring gleich dem anderen String wird eine 0 zurückgegeben.

Beispiel aus plugin_chat (Funktion: HandleSay):

```
	if ( strcasestr(MessageMode[UserIndex], "admin_") >= 0 ) {
	  plugin_exec( MessageMode[UserIndex], Data );
	  return PLUGIN_HANDLED;
```

Es wird ohne Beachtung der Groß- und Kleinschreibung der String „admin_“ im Admin Mod Befehl (in MessageMode[UserIndex]) gesucht. Ist er enthalten, wird der Befehl mit den Parametern (Data) ausgeführt (plugin_exec) und die Ausführung beendet.

Gehört zu: string.inc

Siehe auch: strcasestrx, strstr

8.10.166. strcasestrx

`strcasestrx( sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht einen String (sSearch) in einem anderen String (sSource). Groß- und Kleinschreibung wird nicht beachtet. Wird der Suchstring gefunden, gibt die Funktion eine Zahl größer als 0 zurück. Wenn der String nicht gefunden wurde oder leer ist, gibt die Funktion eine -1 zurück. Ist der Suchstring gleich dem anderen String wird eine 0 zurückgegeben.

Beispiel aus plugin_sdal_logd_hp50[47] (Funktion: HandleSay):

```
        if (strcasestrx(Data, "hp")!=-1){
           if(g_PlayerEnemyID[UserIndex]!=0 && IsDead == 1){
              display_victim(User,UserIndex);
           }else{
              display_statistik(User,UserIndex);
           }
        }else if (strcasestrx(Data, "myhits")!=-1){
           own_statistic(User,UserIndex);
        }else if (strcasestrx(Data, "hits")!=-1){
           most_damage(User,UserIndex);
        }else if (strcasestrx(Data, "weapon")!=-1){
           most_used_weapon(User,UserIndex);
        }
```

Die If-Verzweigung überprüft, ob einige Begriffe in Data vorkommen oder den gleichen Inhalt haben. Ist dies der Fall, werden entsprechende Unterfunktionen aufgerufen, die dem Spieler unterschiedliche Statistikinformationen auf dem Bildschirm darstellen.

Gehört zu: string.inc

Siehe auch: strcasestr, strstr

[47] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_logd_hp50

8.10.167. strcat

`strcat( sDest[], sSource[], iMaxLen );`

`sDest[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Diese Funktion fügt einen String (sSource mit der Länge iMaxLen) an einen anderen (sDest) an.

Beispiel aus plugin_CS (Funktion: LoadDefaultRestrictions):

```
    currentmap(strMap,100);
    strcpy(strName,"WeaponRestrictions_",100);
    strcat(strName,strMap,100);
```

Zunächst wird die aktuelle Map ermittelt und in strMap geschrieben. Danach wird die Zeichenkette „WeaponRestrictions_“ in die Variable strName geschrieben. Abschließend wird der String strName durch den String strMap erweitert. Ist die aktuelle Map „de_dust“, ist strName am Ende „WeaponRestrictions_de_dust“.

Gehört zu: string.inc

Siehe auch: strncat, snprintf

8.10.168. strchr

`strchr( sSource[], iChar );`

`sSource[]`
Typ: String (200)

`iChar`
Typ: Integer (0 - 256)

Die Funktion ermittelt die Position des ersten Vorkommens des Zeichens (iChar) im String (sSource) von links beginnend. Für das Zeichen muss die ASCII-Nummer angegeben werden. Vereinfacht kann auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden. Die Funktion liefert -1 zurück, falls das Zeichen nicht gefunden wurde.

Beispiel aus plugin_blatt_map[48] (Funktion: GetShortName):

```
new pos = strchr(fullname,'_');
if (pos>0) {
  pos = pos +1;
}
else {
  pos = 0;
}
```

Es wird das erste Vorkommen vom Unterstrich im String fullname gesucht. Ist die Position größer als 0, wird die Position inkrementiert. Andernfalls wird die Position auf 0 gesetzt.

Gehört zu: string.inc

Siehe auch: index

[48] http://www.adminmod.de/plugins.php?plugin=plugin_blatt_map

8.10.169. strcmp

`strcmp( sString1[], sString2[] );`

`sString1[]`
Typ: String (200)

`sString2[]`
Typ: String (200)

Die Funktion vergleicht zwei Strings (sString1 und sString2) über ihre gesamte Länge. Groß- und Kleinschreibung wird beachtet. Sind beide Strings identisch, wird eine 0 zurückgegeben. Man erhält eine 0 zurück, wenn beide Strings gleich sind. Ist das erste ungleiche Zeichen in den Strings bei sString1 größer als bei sString2, wird eine Zahl größer als 0 zurückgegeben. Umgekehrt wird eine Zahl kleiner als 0 zurückgegeben.

Beispiel aus plugin_CS (Funktion: SetRestrictions):

```
    } else if(strcmp(Data,"help") == 0) {
        ShowHelp(Status);
```

Falls Data „help“ unter Berücksichtigung der Groß- und Kleinschreibung entspricht, wird eine pluginspezifische Funktion zur Darstellungen der Optionen ausgeführt.

Gehört zu: string.inc

Siehe auch: streq, strncasecmp, strncmp

8.10.170. strcount

`strcount( sSource[], iChar );`

`sSource[]`
Typ: String (200)

`iChar`
Typ: Integer (1 - 256)

Die Funktion zählt das Vorkommen eines Zeichens (iChar) in einem String (sSource). Vereinfacht kann auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden.

Beispiel aus plugin_bk_cron[49] (Funktion: admin_cron_add):

```
if(strcount(task,' ')<6){
   /* selfmessage("Task not valid!"); */
   selfmessage(cron_tnv);
   /* selfmessage("Usage: admin_cron_add <min> <h> <d> <mo> <wd> <cv> <cmd>"); */
   selfmessage(cron_adduse);
   return PLUGIN_HANDLED;
}
```

Die Anzahl der Leerzeichen wird im String task gesucht. Falls die Anzahl kleiner als 6 ist, wird eine zweizeilige Fehlermeldung in der Console des Admins ausgegeben.

Gehört zu: string.inc

[49] http://www.adminmod.de/plugins.php?plugin=plugin_bk_cron

8.10.171. strcpy

`strcpy( sDest[], sSource[], iMaxLen );`

`sDest[]`
Typ: String (200)

`sSource[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Kopiert einen String (sSource) über einen anderen (sDest mit der Länge iMaxLen).

Beispiel aus plugin_base (Funktion: admin_pass):

```
    if (streq(Command, "admin_nopass")==1) {
      strcpy(Data,"^"^"",MAX_DATA_LENGTH);
```

Wenn der ausgeführte Befehl „admin_nopass“ lautet, werden keine Parameter erwartet. Um der Servervariable „sv_password“ später auf einen leeren Wert zu setzen, wird Data auf zwei Anführungszeichen gesetzt (“”).

Gehört zu: string.inc

Siehe auch: strncpy

8.10.172. strcspn

`strcspn( sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht Zeichen aus einem String (sSearch) in einem anderen String (sSource). Sie gibt die Länge des Teilstrings zurück, der von links beginnend, kein Zeichen aus sSearch besitzt. Da Strings Felder sind, ist die Länge gleichzeitig die Position des ersten in sSearch vorhandenen Zeichens.

Beispiel:

```
new sTest[MAX_TEXT_LENGTH]="Test123Test";
new iTest=strcspn(sTest,"1234567890");
```

Im Beispiel wird im String sTest nach Zahlen gesucht werden. Da die erste Zahl an der Position 4 liegt, wird als Länge 4 ausgegeben.

Gehört zu: string.inc

Siehe auch: strspn

8.10.173. streq

`streq( strOne[], strTwo[] );`

`strOne[]`
Typ: String (200)

`strTwo[]`
Typ: String (200)

Die Funktion überprüft unter Berücksichtigung der Groß- und Kleinschreibung, ob zwei Strings (strOne und strTwo) identisch sind. Sind sie identisch wird eine 0, wenn nicht wird eine 1 zurückgegeben.

Beispiel aus plugin_base (Funktion: admin_csay):

```
    if (streq(Color,"red")==1) {
       centersay(Message,10,250,10,10);
```

Wenn der Text Color gleich „red“ ist, wird die Nachricht in Message für 10 Sekunden in einen hellem Rot dargestellt.

Gehört zu: adminlib.inc

Siehe auch: strcasecmp, strmatch

8.10.174. strgsep

`strgsep( sSource[], sDelimiters[], sGrouping[], ... );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`sGrouping[]`
Typ: String (200)

`...`
variable Anzahl an Argumenten und Stringlängen (kommagetrennt)

Diese Funktion trennt einen String (sSource) an im String (sDelimitiers) angegebenen Zeichen. Die Trennung wird dort nicht ausgeführt, wo die Zeichenkette von den im String (sGrouping) angegebenen Zeichen umschlossen wird. Werden nicht ausreichend Argumente (...) für die Aufnahme der Teilstrings angegeben, wird der Rest ungetrennt im letzten Argument übergeben.

Die Funktion gibt zurück, wieviele Teilstrings gefunden wurden bzw. -1, wenn kein Teilstring gefunden wurde.

Beispiel aus plugin_CS (Funktion: SetRestrictions):

```
        strgsep(Data[7]," ","’",sPlayer,MAX_AUTHID_LENGTH,sWhat,MAX_DATA_LENGTH);
        if(check_user(sPlayer) == 0) {
            selfmessage("Unrecognized player: ");
            selfmessage(sPlayer);
            selfmessage("If the player name you specified containes spaces");
            selfmessage("try encosing it with single quotes (’).");
            return PLUGIN_HANDLED;
        }
```

Der String Data[7] wird an den Leerzeichen getrennt. Es werden nur zwei Argumente definiert (sPlayer und sWhat). Falls mehr als ein Leerzeichen auftritt, wird nur am ersten getrennt und der Rest in sWhat geschrieben. Hat ein Spieler ein Leerzeichen im Namen, würde der Spielername getrennt und der zweite Teil des Namens mit in sWhat geschrieben. Daher wurden als Gruppierungszeichen die einfachen Anführungszeichen gewählt. Ist der Spielername von diesen Zeichen umschlossen, wird dieser Teil für die Trennung ausgeschlossen und die Trennung am nächsten Leerzeichen durchgeführt.

Anschließend wird überprüft, ob eine Spieler dieses Namens existiert. Wenn nicht, wird eine vierzeilige Fehlermeldung in der Console ausgegeben.

Gehört zu: string.inc

Siehe auch: strgsplit, strgtok, strgtokrest, strsep, strsplit, strtok, strtokrest

8.10.175. strgsplit

`strgsplit( sSource[], sDelimiters[], sGrouping[], ... );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`sGrouping[]`
Typ: String (200)

`...`
variable Anzahl an Argumenten und Stringlängen (kommagetrennt)

Diese Funktion trennt einen String (sSource) an im String (sDelimitiers) angegebenen Zeichen. Die Trennung wird dort nicht ausgeführt, wo die Zeichenkette von den im String (sGrouping) angegebenen Zeichen umschlossen wird. Werden nicht ausreichend Argumente (...) für die Aufnahme der Teilstrings angegeben, wird der Rest verworfen.

Die Funktion gibt zurück, wieviele Teilstrings gefunden wurden bzw. -1, wenn kein Teilstring gefunden wurde.

Beispiel aus plugin_gnc_filtersay[50] (Funktion: readcfg):

```
for (new i=0;(i<MAX_WORDS && readfile(filename,sData,(i+1),MAX_DATA_LENGTH));i++){
   strgsplit(sData, " ", "^"", BadWord[i], MAX_PL, GoodWord[i], MAX_PL);

}
```

Wenn der Dateiname existiert (fileexists), wird die gesamte Datei zeilenweise ausgelesen. Jede Zeile wird an den Leerzeichen getrennt, wobei in Begriffe in doppelten Anführungszeichen nicht getrennt werden. Ist ein zweites Leerzeichen außerhalb der Anführungszeichen vorhanden, wird der letzte Teilstring ignoriert.

Gehört zu: string.inc

Siehe auch: strgsep, strgtok, strgtokrest, strsep, strsplit, strtok, strtokrest

[50] http://www.adminmod.de/plugins.php?plugin=plugin_gnc_filtersay

8.10.176. strgtok

`strgtok( sSource[], sDelimiters[], sGrouping[], sToken[], iMaxLen );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`sGrouping[]`
Typ: String (200)

`sToken[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Diese Funktion trennt einen String (sSource) am ersten im String (sDelimitiers) angegebenen Zeichen. Die Trennung wird dort nicht ausgeführt, wo die Zeichenkette von den im String (sGrouping) angegebenen Zeichen umschlossen wird. Sollen weitere Teilstrings abgetrennt werden, kann die Funktion wiederholt ausgeführt werden. Dabei darf sSource nicht angegeben werden. Ein möglicher Reststring kann mit strgtokrest ausgegeben werden. Bequemer sind die Funktionen strgsep und strgsplit für mehrere Trennungen.

Die Funktion gibt -1 zurück, wenn kein Trennzeichen gefunden wurde.

Beispiel:

```
new sTest[MAX_TEXT_LENGTH]="?Test/12?/3/Test";
strgtok(sTest,"/","?",sToken1,MAX_DATA_LENGTH);
strgtok("","/","?",sToken2,MAX_DATA_LENGTH);
strgtokrest(sToken3,MAX_DATA_LENGTH);
```

Der String (sTest) soll getrennt an allen Slashes getrennt werden. Dabei sind mit Fragezeichen umrahmte Zeichenketten ausgenommen. Die Aktion wird dann wiederholt und am Ende der Rest ausgegeben. Demnach ist sToken1 = „Test/12“, sToken2 = „3“ und sToken3 =“Test“.

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtokrest, strsep, strsplit, strtok, strtokrest

8.10.177. strgtokrest

`strgtokrest( sRest[], iMaxLen );`

`sRest[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Die Funktion gibt den Reststring (sRest mit der Länge iMaxLen) nach Ausführung von strgtok aus. Ist strgtok oder der Reststring leer, wird -1 zurückgegeben.

Beispiel:

```
new sTest[MAX_TEXT_LENGTH]="?Test/12?/3/Test";
strgtok(sTest,"/","?",sToken1,MAX_DATA_LENGTH);
strgtok("","/","?",sToken2,MAX_DATA_LENGTH);
strgtokrest(sToken3,MAX_DATA_LENGTH);
```

Der String (sTest) soll getrennt an allen Slashes getrennt werden. Dabei sind mit Fragezeichen umrahmte Zeichenketten ausgenommen. Die Aktion wird dann wiederholt und am Ende der Rest ausgegeben. Demnach ist sToken1 = „Test/12“, sToken2 = „3“ und sToken3 =“Test“.

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtok, strsep, strsplit, strtok, strtokrest

8.10.178. strinit

`strinit( sString[] );`

`sString[]`
Typ: String (200)

Die Funktion löscht den Inhalt des angegebenen Strings (sString).

Beispiel aus plugin_hldsld_mapvote (Funktion: ResetVoteData):

```
new i;
for(i=0;i<MAX_MAPS;i++) {
   strinit(Maps[i]);
}
```

Die For-Schleife löscht alle Mapnamen aus dem Feld Maps.

Gehört zu: adminlib.inc

8.10.179. strlen

`strlen( const string[] );`

`const string[]`
Typ: String (200)

Die Funktion gibt die Länge der in einem String (string) abgelegten Zeichenkette aus. Sie kann auch mit gepackten Strings umgehen (siehe dazu strpack).

Beispiel aus plugin_base (Funktion: admin_kick):

```
if (strlen(Reason) != 0) {
   snprintf(Text, MAX_TEXT_LENGTH, "You have been kicked because %s", Reason);
   message(real_user, Text);
}
kick(real_user);
```

Über die Länge des Strings Reason wird überprüft, ob ein Grund für den Kick angegeben wurde. Wenn es der Fall ist, wird dies dem Spieler mitgeteilt (message). Anschließend wird der Spieler vom Server geworfen.

Gehört zu: core.inc

8.10.180. strmatch

`strmatch( sOne[], sTwo[], iLength);`

`strOne[]`
Typ: String (200)

`strTwo[]`
Typ: String (200)

`iLength`
Typ: Integer (1 - 200)

Die Funktion überprüft unter Berücksichtigung der Groß- und Kleinschreibung, ob zwei Strings (strOne und strTwo) für die ersten x Zeichen (iLength) identisch sind. Sind sie identisch wird eine 1, wenn nicht wird eine 0 zurückgegeben.

Beispiel aus plugin_fun (Funktion: HandleSay):

```
    convert_string(HLCommand,Command,MAX_COMMAND_LENGTH);
    convert_string(HLData,Data,MAX_DATA_LENGTH);
    convert_string(HLUserName,User,MAX_NAME_LENGTH);

    strstripquotes(Data);

    if (strmatch(Data, "glow ", strlen("glow "))==1) {
```

Zunächst werden die Funktionsargumente in Small-Strings konvertiert (convert_string). Danach werden mögliche Anführungszeichen um den String entfernt (strstripquotes). Abschließend wird nur dann die If-Verzweigung ausgeführt, wenn die ersten 5 Zeichen von Data mit „glow „ übereinstimmen.

Gehört zu: adminlib.inc

Siehe auch: streq, strcasecmp

8.10.181. strncasecmp

`strncasecmp( sString1[], sString2[], iNum );`

`sString1[]`
Typ: String (200)

`sString2[]`
Typ: String (200)

`iNum`
Typ: Integer (0 - 200)

Die Funktion vergleicht zwei Strings (sString1 und sString2) über die Länge (iNum). Groß- und Kleinschreibung wird nicht beachtet. Sind beide Strings identisch, wird eine 0 zurückgegeben. Man erhält eine 0 zurück, wenn beide Strings gleich sind. Ist das erste ungleiche Zeichen in den Strings bei sString1 größer als bei sString2, wird eine Zahl größer als 0 zurückgegeben. Umgekehrt wird eine Zahl kleiner als 0 zurückgegeben.

Beispiel aus plugin_dio_motm[51] (Funktion: say_motm):

```
if (strncasecmp(strColor,"!red",4)==0) {
   Red = 250;
   Green = 10;
   Blue =10;
```

Es wird überprüft, ob die ersten 4 Zeichen in strColor „!red“ entsprechen. Nur dann werden die Farbanteile (Red, Green und Blue) definiert.

Gehört zu: string.inc

Siehe auch: strcmp, strcasecmp, strncmp, strmatch

[51] http://www.adminmod.de/plugins.php?plugin=plugin_dio_motm

8.10.182. strncat

`strncat( sDest[], sSource[], iNum, iMaxLen );`

`sDest[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

`iNum`
Typ: Integer (0 - 200)

`iMaxLen`
Typ: Integer (0 - 200)

Diese Funktion fügt einen String (sSource mit der Länge iMaxLen) an einen anderen (sDest) an. Es werden jedoch nur eine bestimmte Anzahl an Zeichen (iNum) vom String (sSource) angehängt.

Beispiel:

```
new sTest1[MAX_TEXT_LENGTH]="Test123";
new sTest2[MAX_TEXT_LENGTH]="Test456";
strncat(sTest1,Test2,5,MAX_TEXT_LENGTH);
```

Es werden nur die ersten 5 Zeichen von sTest2 an sTest1 angehängt. Der String sTest1 ist nach der Anwendung von strncat „Test123Test4“.

Gehört zu: string.inc

Siehe auch: strcat, snprintf

8.10.183. strncmp

`strncmp( sString1[], sString2[], iNum );`

`sString1[]`
Typ: String (200)

`sString2[]`
Typ: String (200)

`iNum`
Typ: Integer (0 - 200)

Die Funktion vergleicht zwei Strings (sString1 und sString2) über eine bestimmte Länge (iNum). Groß- und Kleinschreibung wird beachtet. Sind beide Strings identisch, wird eine 0 zurückgegeben. Man erhält eine 0 zurück, wenn beide Strings gleich sind. Ist das erste ungleiche Zeichen in den Strings bei sString1 größer als bei sString2, wird eine Zahl größer als 0 zurückgegeben. Umgekehrt wird eine Zahl kleiner als 0 zurückgegeben.

Beispiel aus plugin_CS (Funktion: SetRestrictions):

```
    if(strncmp(Data,"team ",5) == 0) {
        new sTeam[3];
        new sWhat[MAX_DATA_LENGTH];
        new iTeam;
        strsep(Data[5]," ",sTeam,3,sWhat,MAX_DATA_LENGTH);
```

Falls die ersten fünf Zeichen von Data unter Berücksichtigung der Groß- und Kleinschreibung „team “ entsprechen, wird der String Data[5] über ein Leerzeichen in sTeam und sWhat geteilt.

Gehört zu: string.inc

Siehe auch: strcasecmp, strcmp, strmatch, strncasecmp

8.10.184. strncpy

`strncpy( sDest[], sSource[], iNum, iMaxLen );`

`sDest[]`
Typ: String (200)

`sSource[]`
Typ: String (200)

`iNum`
Typ: Integer (0 - 200)

`iMaxLen`
Typ: Integer (0 - 200)

Kopiert einen String (sSource) über einen anderen (sDest mit der Länge iMaxLen), wobei nur eine bestimmte Anzahl an Zeichen (iNum) kopiert werden.

Beispiel aus plugin_jack9_chime[52] (Funktion: SpeakTime):

```
    if(streq(THour, "12")==1) {
       strncpy(words[1], "twelve ", strlen("twelve "),MAX_DATA_LENGTH);
    }
```

Wenn der String THour „12“ entspricht, werden nur die ersten 7 Zeichen von „twelve „ in words[1] geschrieben.

Gehört zu: string.inc

Siehe auch: strcpy

[52] http://www.adminmod.de/plugins.php?plugin=plugin_jack9_chime

8.10.185. strpack

`strpack( dest[], const source[] );`

`dest[]`
Typ: String (200)

`const source[]`
Typ: String (200)

Durch diese Funktion wird ein String (source) gepackt und in einen anderen (dest) geschrieben. Der neue String benötigt nur ein Viertel des Speicherplatzes. Unter Umständen kann man ein wenig Speicher sparen. Es rentiert sich aber nur, wenn nicht gleichzeitig der gleiche String ungepackt definiert wird. Außerdem sollte man in Admin Mod seine Plugins auf Geschwindigkeit nicht Speicher optimieren. Speicher ist eigentlich nie ein Problem.

Beispiel:

```
new spTest[7 char];
strpack(spTest,"Test123");
```

Es wird ein Feld spTest für einen gepackten String von sieben Zeichen länge definiert. Anschließend wird „Test123“ gepackt und in spTest geschrieben.

Gehört zu: core.inc

Siehe auch: strunpack

8.10.186. strrchr

`strrchr( sSource[], iChar );`

`sSource[]`
Typ: String (200)

`iChar`
Typ: Integer (0 - 256)

Die Funktion ermittelt die Position des ersten Vorkommens des Zeichens (iChar) im String (sSource) von rechts beginnend. Für das Zeichen muss die ASCII-Nummer angegeben werden. Vereinfacht kann auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden. Die Funktion liefert -1 zurück, falls das Zeichen nicht gefunden wurde.

Die Funktion ist ein Alias für rindex.

Beispiel aus plugin_rindy_chasecam[53] (Funktion: plugin_connect):

```
    convert_string(HLIP,UserIP,MAX_IP_LENGTH);
    if(strrchr(UserIP,':') != -1) UserIP[strrchr(UserIP,':')] = 0;
```

Es wird zunächst der HL String HLIP in den Small String UserIP konvertiert. Anschließend wird überprüft, ob der Doppelpunkt in UserIP zu finden ist. Ist dies der Fall wird anstelle des Doppelpunktes ein 0 in die Zelle des Strings geschrieben, was den String an dieser Stelle beendet. Auf diese Weise wird der übergebene Port abgetrennt.

Gehört zu: string.inc

Siehe auch: index, rindex, strchr

[53] http://www.adminmod.de/plugins.php?plugin=plugin_rindy_chasecam

8.10.187. strsep

`strsep( sSource[], sDelimiters[], ... );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`...`
variable Anzahl an Argumenten und Stringlängen (kommagetrennt)

Diese Funktion trennt einen String (sSource) an im String (sDelimitiers) angegebenen Zeichen. Werden nicht ausreichend Argumente (...) für die Aufnahme der Teilstrings angegeben, wird der Rest ungetrennt im letzten Argument übergeben.

Die Funktion gibt zurück, wieviele Teilstrings gefunden wurden bzw. -1, wenn kein Teilstring gefunden wurde.

Beispiel aus plugin_CS (Funktion: SetRestrictions):

```
    if(strncmp(Data,"team ",5) == 0) {
        new sTeam[3];
        new sWhat[MAX_DATA_LENGTH];
        new iTeam;
        strsep(Data[5]," ",sTeam,3,sWhat,MAX_DATA_LENGTH);
```

Falls die ersten fünf Zeichen von Data „team “ unter Berücksichtigung der Groß- und Kleinschreibung entsprechen, wird der String Data[5] über ein Leerzeichen in sTeam und sWhat geteilt.

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtok, strgtokrest, strsplit, strtok, strtokrest

8.10.188. strsplit

`strsplit( sSource[], sDelimiters[], ... );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`...`
variable Anzahl an Argumenten und Stringlängen (kommagetrennt)

Diese Funktion trennt einen String (sSource) an im String (sDelimitiers) angegebenen Zeichen. Werden nicht ausreichend Argumente (...) für die Aufnahme der Teilstrings angegeben, wird der Rest verworfen.

Die Funktion gibt zurück, wieviele Teilstrings gefunden wurden bzw. -1, wenn kein Teilstring gefunden wurde.

Beispiel aus plugin_bk_cron[54] (Funktion: admin_cron_refresh):

```
                strsplit(timevar[j],"/",szaehler,3,snenner,3);
                zaehler=strtonum(szaehler);
                nenner=strtonum(snenner);
```

Der String timevar[j] wird über den Slash als Trennzeichen in zwei Strings (Länge jeweils 3) szaehler und snenner aufgeteilt. Beide Strings werden in Ganzzahlen umgewandelt (strtonum).

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtok, strgtokrest, strsep, strtok, strtokrest

[54] http://www.adminmod.de/plugins.php?plugin=plugin_bk_cron

8.10.189. strspn

`strspn(sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht Zeichen aus einem String (sSearch) in einem anderen String (sSource). Sie gibt die Länge des Teilstrings zurück, der von links beginnend, nur Zeichen aus sSearch besitzt. Da Strings Felder sind, ist die Länge gleichzeitig die Position des ersten in sSearch vorhandenen Zeichens.

Beispiel aus plugin_rindy_advretribution[55] (Funktion: AdminSlapTeam):

```
    if(strlen(Data) == 0 || strlen(Data) != strspn(Data,"1234")) {
        selfmessage("Wrong syntax");
        selfmessage("You must enter the team number (1 - 4).");
```

Es wird überprüft, ob die Stringlänge von Data 0 ist. Weiterhin wird überprüft, wie lang der Teilstring von Data ist, der nur die Zahlen von 1 bis 4 beinhalten. Ist die Länge des Teilstrings ungleich der Gesamtlänge, besteht der String nicht nur aus den Zahlen 1 bis 4. Es wird dann eine zweizeilige Fehlermeldung ausgegeben.

Gehört zu: string.inc

Siehe auch: strcspn

[55] http://www.adminmod.de/plugins.php?plugin=plugin_rindy_advretribution

8.10.190. strstr

`strstr( sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht einen String (sSearch) in einem anderen String (sSource). Groß- und Kleinschreibung wird beachtet. Wird der Suchstring gefunden, gibt die Funktion eine Zahl größer als 0 zurück. Wenn der String nicht gefunden wurde, gibt die Funktion eine -1 zurück. Ist der Suchstring gleich dem anderen String wird eine 0 zurückgegeben.

Beispiel aus plugin_base (Funktion: admin_rcon):

```
if (strstr(Data, "rcon_password") >= 0) {
   reject_message();
   return PLUGIN_HANDLED;
}
```

Hierbei handelt es sich um eine Absicherung, dass kein Admin, der Zugriff zu admin_rcon hat, sich unberechtigt Zugriff zu RCon verschafft, indem er das Passwort ändert. Falls der String Data die Servervariable rcon_password beinhaltet, wird die Meldung „You do not have access to this command“ angezeigt und die weitere Abarbeitung abgebrochen.

Gehört zu: string.inc

Siehe auch: strcasestr, strcasestrx, strstrx

8.10.191. strstripquotes

`strstripquotes( str[] );`

`str[]`
Typ: String (200)

Die Funktion löscht eventuell vorhandene Anführungszeichen am Anfang und am Ende eines Strings. Es ist zwingend notwendig bei der Abarbeitung von Chatnachrichten (say, say_team) diese Funktion auszuführen. Der Text ist immer in Anführungszeichen.

Für beliebige Zeichen am Stringanfang und Ende kann auch strtrim verwendet werden.

Beispiel aus plugin_base (Funktion: admin_dmsg):

```
179     switch( sType[0] ) {
180     case 'i':
181        tType = uid_index;
182     case 's':
183        tType = uid_SessionID;
184     case 'w':
185        tType = uid_wonID;
186     default:
187        tType = uid_invalid;
188     }  // switch()
189
190
191     strstripquotes(sMessage);
192     directmessage( sMessage, iUid, tType );
```

Abhängig vom Buchstaben, der sich in der ersten Zelle von sType befindet, wird der UID Typ festgelegt. Die um Anführungszeichen befreite Nachricht (Zeile 191: strstripquotes) wird der ID (Session ID oder Userindex) geschickt.

Gehört zu: adminlib.inc

Siehe auch: strtrim

8.10.192. strstrx

`strstrx( sSource[], sSearch[] );`

`sSource[]`
Typ: String (200)

`sSearch[]`
Typ: String (200)

Die Funktion sucht einen String (sSearch) in einem anderen String (sSource). Groß- und Kleinschreibung wird beachtet. Wird der Suchstring gefunden, gibt die Funktion eine Zahl größer als 0 zurück. Wenn der String nicht gefunden wurde oder leer ist, gibt die Funktion eine -1 zurück. Ist der Suchstring gleich dem anderen String wird eine 0 zurückgegeben.

Beispiel aus plugin_sdal_allowsounds[56] (Funktion: plugin_init):

```
    new sValue[MAX_TEXT_LENGTH];
    getstrvar( "amv_enable_beta", sValue, MAX_TEXT_LENGTH );
    if(strstrx(sValue,"menu1")!=-1){
       g_menuenabled=1;
    }
```

Der Inhalt der Servervariablen amv_enable_beta wird ausgelesen. Wenn der „menu1“ in sValue gefunden wurde, wird g_menuenabled auf 1 gesetzt.

Gehört zu: string.inc

Siehe auch: strstr, strcasestr, strcasestrx

[56] http://www.adminmod.de/plugins.php?plugin=plugin_sdal_allowsounds

8.10.193. strsubst

`strsubst( sString[], sSubst[], sWith[], iMaxLen );`

`sString[]`
Typ: String (200)

`sSubst[]`
Typ: String (200)

`sWith[]`
Typ: String (200)

`iMaxLen`
Typ: String (1 - 200)

Die Funktion erkennt, wo sich in einem String (sString mit der Länge iMaxLen) ein anderer String (sSubst) befindet. Alle Vorkommen von sSubst werden durch den String sWith ersetzt.

Beispiel aus plugin_retribution (Funktion: RemoveUserFlag):

```
if(get_vaultdata(sAuthID,VaultData,MAX_DATA_LENGTH) != 0) {
   if (strcasestr(VaultData," llama") != -1) {
      strsubst(VaultData," llama", "", MAX_DATA_LENGTH);
      set_vaultdata(sAuthID,VaultData);
   }
}
```

Wenn eine Eintrag mit der AuthID in der vault.ini gefunden wurde, wird überprüft, ob die Zeichenkette „ llama“ enthalten ist. Ist dies der Fall, wird im String VaultData „ llama“ durch eine leere Zeichenkette ersetzt und somit gelöscht. Abschließend wird der Text zurück in die vault.ini geschrieben.

Gehört zu: string.inc

Siehe auch: snprintf

8.10.194. strtok

`strtok( sSource[], sDelimiters[], sToken[], iMaxLen );`

`sSource[]`
Typ: String (200)

`sDelimiters[]`
Typ: String (200)

`sToken[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Diese Funktion trennt einen String (sSource) am ersten im String (sDelimitiers) angegebenen Zeichen. Sollen weitere Teilstrings abgetrennt werden, kann die Funktion wiederholt ausgeführt werden. Dabei darf sSource nicht angegeben werden. Ein möglicher Reststring kann mit strtokrest ausgegeben werden. Bequemer sind die Funktionen strsep und strsplit für mehrere Trennungen.

Die Funktion gibt -1 zurück, wenn kein Trennzeichen gefunden wurde.

Beispiel aus plugin_showip[57] (Funktion: admin_showip):

```
strtok(UserIP[i], ":", IP, MAX_IPADDRESS);
strcat(Name,IP,MAX_NAME_LENGTH);
```

Der String UserIP[i] wird am ersten Doppelpunkt geteilt und der erste Teil in den String IP geschrieben. Anschließend wird dieser an den String Name angehängt.

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtok, strgtokrest, strsep, strsplit, strtokrest

[57] http://www.adminmod.de/plugins.php?plugin=plugin_showip

8.10.195. strtokrest

`strtokrest( sRest[], iMaxLen );`

`sRest[]`
Typ: String (200)

`iMaxLen`
Typ: Integer (0 - 200)

Die Funktion gibt den Reststring (sRest mit der Länge iMaxLen) nach Ausführung von strgtok aus. Ist strgtok oder der Reststring leer, wird -1 zurückgegeben.

Beispiel aus plugin_prison2[58] (Funktion: admin_showip):

```
strtok(line,":",storedmap,MAX_DATA_LENGTH);
strtok("",":",tmp,MAX_DATA_LENGTH);
x = strtonum(tmp);
strtok("",":",tmp,MAX_DATA_LENGTH);
y = strtonum(tmp);
strtok("",":",tmp,MAX_DATA_LENGTH);
z = strtonum(tmp);
strtokrest(tmp,MAX_DATA_LENGTH);
```

Aus dem String line werden über den Doppelpunkt als Trennzeichen Stück für Stück Daten extrahiert und teilweise in Ganzzahlen (x, y, z) umgewandelt. Am Ende wird der Reststring ausgelesen.

Gehört zu: string.inc

Siehe auch: strgsep, strgsplit, strgtok, strgtokrest, strsep, strsplit, strtok

[58] http://www.adminmod.de/plugins.php?plugin=plugin_prison2

8.10.196. strtonum

`strtonum( sString[] );`

`sString[]`
Typ: String (200)

Die Funktion wandelt eine als String vorliegende Zahl (sString) in eine Ganzzahl um.

Beispiel aus plugin_retribution (Funktion: admin_execteam):

```
strbreak(Data, strTeam, Cmd, MAX_TEXT_LENGTH);
if (strlen(Cmd) == 0) {
   selfmessage( "Unparsable format: no command found.");
   return PLUGIN_HANDLED;
}
ExecTeam = strtonum(strTeam);
```

Der String Data wird am ersten Leerzeichen in strTeam und Cmd geteilt. Ist Cmd leer, wird eine Fehlermeldung in der Console ausgegeben. Anderenfalls wird strTeam in eine Ganzzahl umgewandelt.

Gehört zu: admin.inc

Siehe auch: numtostr

8.10.197. strtrim

`strtrim( sString[], sTrim[], iWhere = 2 );`

`sString[]`
Typ: String (200)

`sTrim[]`
Typ: String (200)

`iWhere = 2`
Typ: Integer (0 - 2)

Die Funktion entfernt die in einem String aufgeführten Zeichen (sTrim) am Anfang (iWhere=0), am Ende (iWhere=1) oder am Anfang und Ende (iWhere=2) eines Strings (sString). Sie gibt darüber hinaus die Anzahl der entfernten Zeichen an oder -1, wenn kein Zeichen entfernt wurde.

Beispiel aus plugin_bk_botmanager[59] (Funktion: admin_bot_set):

```
    strbreak(sValue,sBotCvar,sValue,MAX_DATA_LENGTH);
    strtrim(sValue," ",2);
    strtrim(sBotCvar," ",2);
```

Der Wert sValue wird in sBotCvar und sValue aufgeteilt (strbreak). Beiden Variablen werden am Anfang und am Ende von möglichen Leerzeichen befreit.

Gehört zu: string.inc

Siehe auch: strstripquotes

[59] http://www.adminmod.de/plugins.php?plugin=plugin_bk_botmanager

8.10.198. strunpack

`strunpack( dest[], const source[] );`

`dest[]`
Typ: String (200)

`const source[]`
Typ: String (200)

Durch diese Funktion wird ein String (source) entpackt und in einen anderen (dest) geschrieben. Die meisten Funktionen benötigen ungepackte Strings.

Unter Umständen kann man ein wenig Speicher sparen. Es rentiert sich aber nur, wenn nicht gleichzeitig der gleiche String ungepackt definiert wird. Außerdem sollte man in Admin Mod seine Plugins auf Geschwindigkeit nicht Speicher optimieren. Speicher ist eigentlich nie ein Problem.

Beispiel:

```
new spTest[7 char]=!"Test123";
new sTest[7];
strunpack(sTest,spTest);
```

Es wird ein gepackter String der Länge 7 definiert. Es ist das Ausrufezeichen vor der Zeichenkette „Test123“ zu beachten, das den Compiler anweist den Text als gepackt zu handhaben. Zu Letzt wird der String spTest in sTest entpackt.

Gehört zu: core.inc

Siehe auch: strpack

8.10.199. swapchars

`swapchars( c );`

`c`
Typ: Integer (-2147483648 bis 2147483647)

Die Funktion dreht die Bytes in einem Wert (c) um.

```
new iTest=33;
iTest=swapchars( iTest );
```

Als zu drehender Wert wird 33 verwendet. Es handelt sich dabei um das Bitmuster „00000000000000000000000000100001“. Als Ergebnis wird wieder in iTest das Bitmuster „10000100000000000000000000000000“ geliefert (entspricht 553648128). Anhand des Bitmusters kann man die Funktionsweise von swapchars erkennen.

Gehört zu: core.inc

8.10.200. systemtime

`systemtime( );`

Die Funktion liefert die Sekunden seit dem 1.1.1970 zurück.

Beispiel aus plugin_retribution (Funktion: admin_gag):

```
    if (strlen(strTime) != 0) {
       new Time = systemtime();
       GagTime = strtonum(strTime) * 60;
       GagTime += Time;
    }
```

Wenn der String strTime nicht leer ist, wird die aktuelle Sekundenzahl ermittelt. Anschließend wird der String strTime in eine Zahl umgewandelt und mit 60 multipliziert. Abschließend wird die Sekundenzahl zur Variable GagTime addiert.

Gehört zu: admin.inc

Siehe auch: servertime

8.10.201. teleport

`teleport( sPlayer[], iX, iY, iZ );`

`sPlayer[]`
Typ: String (33)

`iX`
Typ: Integer (-2147483648 - 2147483647)

`iY`
Typ: Integer (-2147483648 - 2147483647)

`iZ`
Typ: Integer (-2147483648 - 2147483647)

Diese Funktion teleportiert einen Spieler (sPlayer) zu den angegebenen Koordinaten (iX, iY, iZ). Wenn admin_fx in der adminmod.cfg aktiviert wurde, gibt es zusätzliche Effekte.

Valide Koordinaten kann man über die Funktion get_userorigin erhalten.

Beispiel aus plugin_retribution (Funktion: admin_bury):

```
get_userorigin(TargetName, x, y, z);
teleport(TargetName, x, y, (z-25));
```

Zunächst werden die Koordinaten des Spielers (TargetName) ermittelt (get_userorigin). Anschließend wird er um 25 Einheiten tiefer gesetzt. Sofern sich der Spieler zu diesem Zeitpunkt nicht hoch in der Luft befindet, wird er in den Boden versetzt, so dass er sich nicht mehr bewegen kann.

Gehört zu: admin.inc

Siehe auch: get_userorigin

8.10.202. timeleft

```
timeleft( iPrintToConsole = 1 );
```

`iPrintToConsole = 1`
Typ: Integer (0 - 1)

Die Funktion ermittelt die verbleibenden Sekunden bis zum Mapwechsel. Über das Argument iPrintToConsole wird eingestellt, ob die Restzeit in Minuten und Sekunden auch in der Console angezeigt wird, wobei 0 keine Anzeige bedeutet.

Beispiel aus plugin_chat (Funktion: SayTimeleft):

```
SayTimeleft() {
    new Text[MAX_TEXT_LENGTH];
    new Seconds = timeleft(0);

    Seconds /= 60;
    snprintf(Text, MAX_TEXT_LENGTH, "Time remaining on map: %i minutes", Seconds);
    say(Text);
}
```

Die Anzahl der bis zum Mapende verbleibenden Sekunden wird ermittelt und in die Variable Seconds geschrieben. Diese wird durch 60 geteilt, um Minuten zu erhalten. Die Minutenzahl wird in den String Text eingebettet, der abschließend im Chat ausgegeben wird.

Gehört zu: admin.inc

Siehe auch: maptime

8.10.203. tolower

`tolower( c );`

`c`
Typ: Integer (0 - 256)

Wenn es sich bei einem Zeichen (c) um einen Großbuchstaben handelt, wandelt die Funktion dieses in einen Kleinbuchstaben um. Um nicht den ASCII-Code verwenden zu müssen, kann vereinfacht auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden.

Beispiel aus plugin_CS (Funktion: strtolower):

```
strtolower(String[]) {
    new i;
    for(i=0;String[i];i++) {
        String[i] = tolower(String[i]);
    }
}
```

Diese Funktion wandelt über ein For-Schleife alle Zeichen des Strings String in Kleinbuchstaben um.

Gehört zu: core.inc

Siehe auch: toupper

8.10.204. toupper

`toupper( c );` Typ: Integer (0 - 256)

Wenn es sich bei einem Zeichen (c) um einen Kleinbuchstaben handelt, wandelt die Funktion dieses in einen Großbuchstaben um. Um nicht den ASCII-Code verwenden zu müssen, kann vereinfacht auch das entsprechende Zeichen in einfache Anführungszeichen gesetzt werden.

Beispiel aus plugin_CS (Funktion: ShowClass):

```
    Message[0] = toupper(Message[0]);
    selfmessage(Message);
```

Diese Funktion wandelt den ersten Buchstaben des Strings Message in einen Großbuchstaben, um ihn dann auf der Console des Admins auszugeben.

Gehört zu: core.inc

Siehe auch: tolower

8.10.205. typesay

`typesay( sMessage[], iTime, iRed, iGreen, iBlue );`

`sText[]`
Typ: String (500) (max. Zeilenlänge: 80)

`iTime`
Typ: Integer (0 - 2147483647)

`iRed`
Typ: Integer (0 - 255)

`iGreen`
Typ: Integer (0 - 255)

`iBlue`
Typ: Integer (0 - 255)

Mit dieser Funktion kann man eine bunte Nachricht (sText) für alle Spieler in der linken unteren Ecke des Bildschirms produzieren. iTime ist die Einblendzeit in Sekunden. iRed ist der Rotanteil, iGreen der Grünanteil und iBlue der Blauanteil in der Nachricht. Die Funktion messageex ermöglicht die gleiche Funktionalität jedoch ohne Farbwahl für einzelne Spieler.

Beispiel aus plugin_base (Funktion: admin_tsay):

```
	if (streq(Color,"red")==1) {
	   typesay(Message,10,250,10,10);
```

Wenn der Text Color gleich „red“ ist, wird die Nachricht in Message für 10 Sekunden in einen hellem Rot dargestellt. Es ist zu beachten, dass eine Zeile maximal 80 Zeichen lang sein darf. Mit Zeilenumbrüchen sind bis zu 500 Zeichen möglich.

Gehört zu: admin.inc

Siehe auch: centersay, centersayex, messageex

8.10.206. unban

`unban( sWONID[] );`

`sWONID[]`
Typ: String (39)

Die Funktion entbannt den Spieler unter Angabe der Steam ID oder IP (sWONID).

Beispiel aus plugin_base (Funktion: admin_unban):

```
public admin_unban(HLCommand,HLData,HLUserName,UserIndex) {
   new Command[MAX_COMMAND_LENGTH];
   new Data[MAX_DATA_LENGTH];
   new User[MAX_NAME_LENGTH];

   convert_string(HLCommand,Command,MAX_COMMAND_LENGTH);
   convert_string(HLData,Data,MAX_DATA_LENGTH);
   convert_string(HLUserName,User,MAX_NAME_LENGTH);
   unban(Data);
   log_command(User,Command,Data);
   return PLUGIN_HANDLED;
}
```

Alle HL Strings werden in Small Strings umgewandelt. In der Variable Data wird die Steam ID oder die IP erwartet. Abschließend wird die Aktion basierend auf admin_quiet in die Logdateien geschrieben.

Gehört zu: admin.inc

Siehe auch: ban

8.10.207. userlist

`userlist( sPattern[]= "" );`

`sPattern[]`
Typ: String (33)

Diese Funktion gibt in der Console des aufrufenden Admins einen Überblick über die momentan Spielenden. Dabei wird der Name, die Session ID, die Steam ID, der Accesslevel und der Immunitätsstatus des Spielers angegeben. Man kann auch die Spielerausgabe über ein optionales Muster einschränken.

Beispiel aus plugin_base (Funktion: admin_userlist):

```
public admin_userlist(HLCommand,HLData,HLUserName,UserIndex) {
   new Data[MAX_DATA_LENGTH];

   convert_string(HLData,Data,MAX_DATA_LENGTH);
   userlist(Data);
   return PLUGIN_HANDLED;
}
```

Es wird die Spielerliste ausgegeben, die ggf. durch ein Muster (z.B. „a“ für alle Spieler, die ein „a“ im Namen haben) eingeschränkt wurde.

Gehört zu: admin.inc

8.10.208. valid_map

`valid_map( sMap[] );`

`sMap[]`
Typ: String (100)

Die Funktion überprüft, ob sich eine angegebene Map (sMap) auf dem Server befindet. Ist eine maps.ini in der adminmod.cfg definiert, gelten nur die dort eingetragenen Maps als gültig. Dieser Check lässt sich mit valid_mapex umgehen.

Beispiel aus plugin_base (Funktion: admin_map):

```
    if (valid_map(Data)==1) {
       say_command(User,Command,Data);
       changelevel(Data, 4);
```

Falls es sich bei Data um eine gültige Map (auch unter Berücksichtigung der maps.ini) handelt, wird abhängig von der Einstellung admin_quiet eine Nachricht in den Chat geschrieben und nach vier Sekunden ein Wechsel auf die Map initiiert.

Gehört zu: admin.inc

Siehe auch: valid_mapex

8.10.209. valid_mapex

`valid_mapex( sMap[] );`

`sMap[]`
Typ: String (100)

Die Funktion überprüft, ob sich eine angegebene Map (sMap) auf dem Server befindet. Ein zusätzlicher Check, ob die Map auch in der maps.ini steht, kann mit valid_map durchgeführt werden.

Beispiel:

```
if (valid_mapex(Data)==1) {
   say_command(User,Command,Data);
   changelevel(Data, 4);
```

Falls es sich bei Data um eine gültige Map (unter Ausschluss der maps.ini) handelt, wird abhängig von der Einstellung admin_quiet eine Nachricht in den Chat geschrieben und nach vier Sekunden ein Wechsel auf die Map initiiert.

Gehört zu: admin.inc

Siehe auch: valid_map

8.10.210. version

`version( );`

Diese Funktion gibt die aktuelle Admin Mod Version in der Spielerconsole aus.

Beispiel:

```
public admin_ver(HLCommand,HLData,HLUserName,UserIndex) {
   version();
   return PLUGIN_HANDLED;
}
```

Es wird die Versionsnummer Admin Mods beim aufrufenden Admin in der Console ausgegeben.

Gehört zu: admin.inc

8.10.211. vote

`vote( sVoteString[], ... );`

`sVoteString[]`
Typ: String (500)

`...`
variable Anzahl (max. 12) an Argumenten (kommagetrennt)

Diese Funktion führt einen Vote aus. Dabei muss eine Frage gestellt werden (sVoteString). Anschließend folgen kommagetrennt, die möglichen Anworten (max. 10). Die letzten zwei Argumente beinhalten die nach dem Vote aufzurufende Funktion und die an diese Funktion zu übergebenen Parameter. Mehr zur Verwendung dieser Funktion ist dem Tutorial zu entnehmen.

Beispiel aus plugin_base (Funktion: admin_vsay):

```
    if (vote_allowed()!=1) {
       selfmessage( "Vote not allowed at this time.");
       return PLUGIN_HANDLED;
    }

    log_command(User,Command,Data);
    vote(Data,"Yes","No","HandleVsay", Data);
    return PLUGIN_HANDLED;
```

Zunächst wird überprüft, ob überhaupt ein Vote erlaubt ist (vote_allowed). Wenn dies nicht der Fall ist, wird dem Admin eine Fehlermeldung ausgegeben. Anderenfalls wird der Befehl in die Logdateien geschrieben und ein Vote mit der in Data stehenden Frage und den Antwortmöglichkeiten „Yes“ und „No“ ausgeführt. Die aufzurufende Funktion lautet dabei „HandleVsay“, und die Frage wird als Parameter an diese Funktion übertragen.

Gehört zu: admin.inc

Siehe auch: vote_allowed

8.10.212. vote_allowed

`vote_allowed( );`

Diese Funktion überprüft, ob ein Vote ausgeführt werden darf. Sie gibt 1 zurück, wenn ein Vote erlaubt ist. Anderenfalls gibt sie 0 zurück.

Beispiel aus plugin_base (Funktion: admin_vsay):

```
if (vote_allowed()!=1) {
   selfmessage( "Vote not allowed at this time.");
   return PLUGIN_HANDLED;
}

log_command(User,Command,Data);
vote(Data,"Yes","No","HandleVsay", Data);
return PLUGIN_HANDLED;
```

Zunächst wird überprüft, ob überhaupt ein Vote erlaubt ist (vote_allowed). Wenn dies nicht der Fall ist, wird dem Admin eine Fehlermeldung ausgegeben. Anderenfalls wird der Befehl in die Logdateien geschrieben und ein Vote mit der in Data stehenden Frage und den Antwortmöglichkeiten „Yes“ und „No“ ausgeführt.

Gehört zu: admin.inc

Siehe auch: vote

8.10.213. writefile

`writefile( sFilename[], sLine[], iLineNum = -1 );`

`sFilename[]`
Typ: String (200)

`sLine[]`
Typ: String (200)

`iLineNum`
Typ: Integer (0 - 2147483647)

Die Funktion schreibt einen Text (sLine) in eine Zeile (iLineNum) einer Datei (sFilename). Wird iLineNum nicht oder als -1 angegeben, wird der Text als neue Zeile an die Datei angehängt.

Die Funktion ist sehr zeitintensiv, da gewartet werden muss bis der Festplattenzugriff erfolgt ist. Schreibt man mehrere Zeilen mit einer For-Schleife, wird bei jeder Zeile die Datei neu geöffnet, was weitere Verzögerungen verursacht. Man sollte sich daher genau überlegen, ob man diese Funktion nutzen möchte. Beim Schreiben während des Serverstarts bzw. beim vereinzelten Anhängen von Text ist dies noch akzeptabel, ein Schreiben während der Spielzeit kann bei einer For-Schleife durchaus zu Aussetzern im Spiel führen (Lag). Wenn möglich sollte man seine Einstellungen über die vault.ini lesen und schreiben.

Beispiel aus plugin_bk_res[60] (Funktion: add_res_data):

```
	iLines=filesize(sMap,lines);
	for(i=1;i<=iLines;i++){
	   readfile(sMap,sLine,i,MAX_DATA_LENGTH);
	   writefile(sMap2,sLine,-1);
	}
```

Es wird die Anzahl der Zeilen der Datei (sMap) ausgelesen. Anschließend werden die Daten über eine For-Schleife zeilenweise von einer Datei (sMap) in eine andere (sMap2) kopiert. Die Zeilen übernehmen nicht unbedingt die Zeilennummer der Ursprungsdatei. Die Zeilen werden beim Schreiben nur angehängt (-1).

Gehört zu: admin.inc

Siehe auch: deletefile, filesize, readfile, resetfile

[60] http://www.adminmod.de/plugins.php?plugin=plugin_bk_res

8.11. Compiler Fehler und Warnungen

Niemand arbeitet fehlerfrei. Einige Tipp- oder auch Denkfehler sorgen für Compilerfehler oder -warnungen. U.U. kann man auch den Compiler zum Absturz bringen. Oftmals sind die Rückmeldungen auf den ersten Blick nicht besonders verständlich.

Die Syntax der Fehlermeldungen lässt sich am besten an Hand eines Beispiels erklären:

`plugin_test.sma(13) Warning [217]]: loose indentation`

Zunächst wird angegeben, in welchem Quellcode der Fehler auftritt. In den runden Klammern wird die Zeile angegeben, in der der Fehler aufgefallen ist. Meistens muss man sich die vorhergehende Zeile ansehen. Insbesondere bei Klammersetzungsfehlern im Quellcode kann der Fehler aber auch an einer gänzlich anderen Stelle oberhalb der gemeldeten liegen.

Es wird weiterhin angegeben, ob es sich um einen Fehler oder nur eine Warnung handelt. Bei Warnungen wird zwar der Compiliervorgang weitergeführt, ob dieser aber auch fehlerfrei durchgeführt wurde, ist nicht garantiert. Es gibt keinen Grund eine Warnung zu ignorieren. Jedes saubere Plugin muss sich warnungsfrei compilieren lassen. In eckigen Klammern ist die Fehler- bzw. Warnungsnummer gegeben, an die sich die ausgeschriebene Fehlermeldung anschließt.

Die folgenden Meldungen wird man häufiger finden:

Error [001] Die Meldung „expected token: “;”, but found “-identifier-”“ deutet darauf hin, dass am Ende der vorhergehenden Zeile ein Semikolon vergessen wurde.

Error [010] Der Hinweis auf „invalid function or declaration“ hat meistens den Grund, dass man sich in der angegebenen Zeile vertippt hat.

Error [021] Die Meldung „symbol already defined: “set_timer”“ bedeutet, dass man versucht, eine bereits definierte Funktion neu zu definieren.

Error [029] Die Nachricht „invalid expression, assumed zero“ deutet auf ein Klammersetzungsproblem hin.

Error [039] Die Meldung „argument type mismatch (argument 1)“ bedeutet, dass einer Funktion eine Variable mit falschem Datentyp übergeben wurde (z.B. Integer statt String). Als Hilfestellung wird auch angegeben, bei welchem Argument dies geschehen ist.

Warning [209] Die Meldung „number of arguments does not match definition“ sagt aus, dass man einer Funktion zu wenige oder zu viele Argumente übergeben hat.

Warning [209] Beim Hinweis „symbol is never used: “test”“ wird darauf hingewiesen, dass eine als privat definierte Funktion von keiner anderen Funktion aus dem Plugin aufgerufen wird. Der Code sollte dann entfernt oder die Funktion „public“ gemacht werden.

Warning [215] Der Ausdruck „expression has no effect“ meint meistens, dass in der

Zeile ein Vergleich nur mit einem Gleichheitszeichen angestellt wurde („=“ statt „==“).

Warning [217] Die Meldung „loose indentation“ sollte bei konsequenter Anwendung von Klammern nicht mehr auftreten. Man kann aber auch auf Klammern verzichten und Tabulatoren nutzen. Dafür müssen dann aber auch die Einrückungen komplett stimmen. Auf Grund der besseren Übersicht und der geringeren Fehlerträchtigkeit sind aber Klammern zu empfehlen.

Warning [219] Wenn die Meldung „local variable identifier shadows a symbol at a preceding level“ erscheint, versucht man eine Variable zu definieren, die bereits global genutzt wird. Meistens wurde sie in einer der Includes definiert und ist daher nicht gleich ersichtlich. Es sollte ausreichen, den Variablennamen zu ändern.

Der Compiler kann aber auch ganz unvermutet abstürzen. Dies kann in den folgenden Fällen auftreten:

- Doppelt definierte globale Variablen
- Irtümlich doppelt definierte Variablen. Eine Variable darf maximal 31 Zeichen lang sein. Ist sie länger als 31 Zeichen, wird sie abgeschnitten. Sind dann zwei Variablen in den ersten 31 Zeichen gleich, führt das zum Absturz.
- Wenn bei großen Plugins in einer der ersten Zeilen ein abschließendes Anführungszeichen vergessen wurde, kann dies zum Absturz führen.

8.12. Runtime Fehler

Neben den Fehlern beim Compilieren können auch Fehler während der Laufzeit des Plugins auftreten. Die Meldungen können dann den Logdateien entnommen werden.

3 - AMX_ERR_STACKERR Dem Plugin ist der Speicher für lokale Variablen ausgegangen (Standard: 8 kB). Mittels der Definition `#pragma dynamic 4096` kann man den Speicher auf 16 kB (4096 Cells) erhöhen. Dies erfordert in jedem Fall ein Neucompilieren des Plugins.

4 - AMX_ERR_BOUNDS Es wurde versucht auf eine Zelle eines Feldes zuzugreifen, das nicht existiert. Meistens ist eine For-Schleife einen Schritt zu lang gewählt worden.

10 - AMX_ERR_NATIVE Einer Admin Mod Funktion wurden ungültige Argumente (z.B. außerhalb des Gültigkeitbereichs) übergeben. Dies kann auch passieren, wenn man eine Zeile aus einer Datei liest (readfile), die länger als die maximal definierte Stringlänge ist.

11 - AMX_ERR_DIVIDE Bei einer Berechnung wurde versucht durch Null zu teilen.

17 - AMX_ERR_FORMAT Beim Plugin handelt es sich nicht um ein gültiges. Als

Gründe kommen in Frage, dass man die SMA-Datei hochgeladen hat, ein fehlerhaftes Plugin oder aber auch ein AMXMod(X) Plugin versucht zu laden.

19 - AMX_ERR_NOTFOUND Ein Befehl wurde in Admin Mod definiert und auf eine öffentliche Funktion verwiesen, die von Admin Mod aber nicht gefunden wurde. Mögliche Gründe dafür können eine fehlende „public“ Deklaration oder ein Tippfehler beim Funktionsnamen sein.

122 - AMX_ERR_INIT Ein interner Admin Mod Fehler ist aufgetreten. Einziger bekannter Grund ist der Inkrementierungsversuch bei einer Feldzelle mittels „++“, was auf Grund eines Compilerbugs fehlschlägt. Oft stürzt der Server aber auch einfach sang- und klanglos ohne Fehlermeldung ab.

Zum Komplettabsturz des Servers können auch Endlosschleifen führen. Auch sollte man vermeiden Variablen direkt in Schleifen zu definieren.

```
for(new i=1;i<10;i++) {
  say(``Test'');
}
```

Die Variablen werden dann nicht immer im Speicher freigegeben. Inbesondere in Kombination mit einer „break“ Anweisung kann es zu einem Absturz kommen. Man sollte daher stets die Variablen vor der Schleife definieren.

8.13. LogD Events

Wie im Tutorial gezeigt, kann man die in Admin Mod vorhandenen Events durch LogD erweitern. Jeder LogD-Event hat eine bestimmte Nummer und eine dafür festgelegte Rückgabesyntax, die als Gesamtstring zurückgegeben wird. Der String muss im Plugin an den Leerzeichen getrennt und ggf. in Integer umgewandelt werden. Im Folgenden sollen die zur Verfügung stehenden Events mit ihrer Rückgabesyntax aufgelistet werden (Tabelle 8.1).

Die Variablen werden in der angegebenen Reihenfolge mit Leerzeichen als Trennzeichen im Rückgabestring zurückgegeben. Die Bezeichnung der Variablen ist in diesem Fall willkürlich. Das „s“ deutet an, dass der Wert als String in Admin Mod ausgewertet werden soll. Entsprechend steht das „i“ für einen Zahlenwert. Der eigentliche Name der Variablen soll den Zweck andeuten. „UI“ steht dabei verkürzt für den Userindex.

Um Leerzeichen (z.B. beim Chat) darstellen zu können, ohne dass diese beim Trennen Ärger bereiten, werden diese durch „$“ ersetzt. Optionen im String werden von ihrem Wert mit einem „#“ abgesetzt.

Tabelle 8.1.: LogD Events

Event	Name	Variablen	Beispiel
5	Servername	sHostname	Mein$Server
50	Connect	iPlayerUI iPlayerIP	12 1.2.3.4:12345
51	Enter Game	iPlayerUI	16
52	Disconnect	iPlayerUI	11
53	Suicide	iPlayerUI sWaffe	9 grenade
54	Team Select	iPlayerUI sTeam	5 TERRORIST
55	Role Select	iPlayerUI sRole	2 Medic
56	Change Name	iPlayerUI sNewName	6 Me$Admin
57	Kill	iKillerUI iVictimUI sWaffe	6 4 p228
58	Damage	iAttackerUI iVictimUI damage#iDamage damage_armor#iDA health#iHealth armor#iArmor	3 1 p228 damage#28 damage_armor#0 health#72 armor#0
59	PvP Action	iInvokerUI iReceiverUI sAction	5 10 Medic_Heal
60	Player Action	iPlayerUI sAction	1 Planted_The_Bomb
61	Team Action	sTeam sAction CT#iScoreCT T#iScoreT	CT Target_Saved CT#1 T#0
62	World Action	sAction	Round_Start
63	Chat	n/t (Normal/Team) iPlayerUI sText	n 7 WeristAdmin?
64	Team Alliance	sInvokerTeam sReceiverTeam	Red Blue
65	Team Score Report	sTeam iScore iNumTeamPlayer	TERRORIST 9 7
66	Private Chat	iInvokerUI iReceiverUI sText	5 3 WeristAdmin?

8.14. Properties in Small

Als Schlüssel kann entweder der Name oder der Wert der Property benutzt werden. Properties sind Schlüssel-Wert-Paare, die in der AMX-Scriptengine gespeichert werden. Jede Property hat eine Zeichenkette und eine Ganzzahl. Welches davon der Schlüssel, und welches der Wert ist, kann der Scriptautor bestimmen, indem er einen der beiden Parameter leer lässt. Die ID der Property dient dazu, dass verschiedene Scripte ihre Propertys auseinander halten können. Eine Property wird durch ihre ID und ihren Schlüssel eindeutig identifiziert.

Die Vorteile von Properties gegebnüber globalen Variablen sind, dass der Speicherplatz dynamisch verwaltet wird, und somit praktisch unbegrenzt Daten gespeichert werden können. Weiterhin bleiben die Daten auch über einen Kartenwechsel hinaus erhalten. Die Nachteile sind, dass mit zunehmender Benutzung von Properties die Zugriffsgeschwindigkeit abnimmt, da diese als einfach verkettete Liste implementiert sind. Ausserdem gibt es keine Möglichkeit festzustellen, ob im Propertybereich einer ID bereits Werte liegen. Zuletzt obliegt es dem Script-Autor, die angelegten Properties wieder zu löschen, da ansonsten der Speicherverbrauch immer weiter anwächst. Properties sollten in Adminmod daher z.Z. nicht verwendet werden.

Es wird empfohlen stattdessen auf die Befehle zur Vault Datei (vault.ini) zurückzugreifen (z.B. get_vaultdata). Wer dennoch die Properties nutzen möchte, kann die Funktionen deleteproperty, existproperty, getproperty und setproperty verwenden.

A. Besondere Customplugins

A.1. Plugin_sank_sounds

Das Plugin_sank_sounds reagiert auf vorher definierte Chateingaben mit der Ausgabe eines zugeordneten Sounds. Außerdem reagiert es auf das Connecten eines Spielers auf den Server, sowie auf einen den Server verlassenden Spieler.

A.1.1. Konfiguration

Folgende Einstellungen sind in der adminmod.cfg vorzunehmen, damit das Plugin funktioniert:

```
file_access_read 1
file_access_write 1
```

Um korrekt zu funktionieren, müssen die Sounds (Wav-Dateien) auf dem Server im richtigen Verzeichnis bereit gestellt werden, z.B. /cstrike/sound/misc. Weiterhin muss gewährleistet sein, dass ein Clientrechner sich diese Sounddateien beim Connecten herunterladen kann. Das erreicht man durch die Erstellung von Resolve (Res) Dateien.

Plugin_sank_sound besitzt eine Konfigurationsdatei, in der definiert wird, auf welche Chateingaben das Plugin mit welchem Sound reagieren soll. Diese Datei heisst SND-LIST.CFG und gehört direkt in die Mod Directory, z.B. /cstrike.

Die Einträge darin sind, wie folgt, angeordnet:

```
Chateingabe; misc/sounddatei.wav
```

Eine SND-LIST.CFG könnte folgendermaßen aussehen:

```
# TimeStamp: 20:38 December 28, 2001
# File created by: Admin
# Important parameters:
SND_KICK; 20
SND_WARN; 17
SND_JOIN; sank\theone.wav
SND_EXIT; misc\comeagain.wav
SND_DELAY; 0
SND_SPLIT; 0
```

```
EXACT_MATCH; 1

# Word/Wav combinations:
# The '@' symbol signifies an admin-only sound
woohoo; misc\woohoo.wav;misc\woohoo2.wav
crap; misc\awwcrap.wav;misc\awwcrap2.wav;misc\awwman.wav
ha ha; misc\haha.wav
doh; misc\doh.wav;misc\doh2.wav;misc\doh3.wav;@misc\dohoo.wav;
bud; misc\bud.wav;misc\weis.wav;misc\er.wav
ouch; misc\ow.wav
weis; misc\weis.wav
@doomed; misc\doomed.wav
```

Die Chateingabe: ha ha verursacht also das Abspielen der Datei „misc\haha.wav“. Mit dem Zeichen @ wird ein Sound, oder sogar ein Schlüsselwort für authorisierte Admins reserviert. Die Pfadangabe des Verzeichnisses /sound wird in der SND-LIST.CFG weggelassen.

Weiterhin sind noch die Angaben zu SND_JOIN (Sound der beim Connecten abgespielt wird) und SND_EXIT (Sound der beim Verlassen eines Spielers abgespielt wird) zu erwähnen.

Das Plugin liesst bis zu 40 Schlüsselworte aus der SND-LIST.CFG aus. Mehr Schlüsselworte zu definieren macht also keinen Sinn, wenn man nicht den Quellcode des Plugins verändert. Für einen Server im Internet ist diese Anzahl von Worten, wenn man die Zeit berücksichtigt, die zum Connecten benötigt wird, mehr als genug.

Dennoch wird häufig der Wunsch nach mehr als 40 Sounds laut. Dies war dann zumeist für LAN Server angedacht. Mehr Sounds zu definieren ist auch ohne große Probleme möglich. Dazu muss die Datei plugin_sank_sound.sma mit einem geeigneten Editor geöffnet werden, z.B. Notepad. In der Zeile 58 des Quellcodes befindet sich die Zeile:

```
#define MAX_KEYWORDS 40 // Maximum number of keywords
```

Diese Zeile sagt aus, dass maximal 40 Schlüsselworte aus der SND-LIST.CFG ausgelesen werden. Wenn man 150 Schlüsselworte auslesen will, so ändert man diese Zeile in

```
#define MAX_KEYWORDS 150 // Maximum number of keywords
```

Danach muss die Datei compiliert und ggf. konvertiert werden.

A.1.2. Probleme

Sind keine Soundausgaben zu hören, wurden zumeist die Sounds nicht auf den Server kopiert bzw. in ein falsches Verzeichnis. Es kann auch sein, dass keine .res Dateien angelegt wurden. Als letzte Möglichkeit kommt in Betracht, dass die Datei SND-LIST.CFG nicht an der Stelle ist, an der sie erwartet wird (Moddirectory z.B. /cstrike)

Werden nicht alle Sounds abgespielt, kann es sein, dass mehr als 40 Schlüsselworte ange-

legt wurden, jedoch vergessen wurde, den Quellcode des Plugins anzupassen. Alternativ wurde das Format (Schlüsselwort; Pfad/Wavdatei.wav) falsch eingegeben.

A.2. plugin_logd_ffmon

Das Plugin_logd_ffmon reagiert auf Teamkills (TK) und Teamattacken (TD), die bei Counter-Strike unerwünscht sind. Es zeichnet sich durch einen großen Umfang und eine reichhaltige Funktionspalette aus.

- Reaktion auf Teamkills (TK) und Teamattacken (TD).
- Gibt dem Opfer eines TK mehrere Möglichkeiten auf den TK zu reagieren oder zu vergeben (auch mittels Menü bedienbar).
- Ab einer bestimmten TK-Anzahl wird der TKer vom Server gebannt.
- Ab einer bestimmten TD-Anzahl wird dieser als unvergebener TK gewertet.
- Verschärfte Bestrafung kurz nach dem Rundenstart
- Speichert TK über einen langen Zeitraum
- Bei unterschiedlich starken Teams besteht Möglichkeit zum Mapwechsel nach festgelegter Rundenzahl.
- Admin-Immunität möglich
- Spielstandsanzeige am Rundenende
- Mapvote nach einer bestimmten Rundenanzahl möglich

Das Script ist aus der Konsole heraus voll konfigurierbar. Es greift dabei auf die Datei vault.ini zurück. Aufgrund seines Umfanges und seiner Flexibilität ist es jedoch oftmals falsch konfiguriert und funktioniert dann nicht wunschgemäß.

Dem Opfer eines TKs steht je nach Einstellung entweder ein Menü zum Bestrafen, bzw. Vergeben zur Verfügung, oder er gibt ein entsprechendes Wort in den Chat ein.

`forgive, bury, slap, slay, kick, ban` oder `chicken`

Reagiert das Opfer nicht, so gilt der TK als unvergeben. Besteht die Möglichkeit eines Banns, so kann dieser im Falle der Ungerechtfertigtheit durch die Chateingabe: `overrideban` aufgehoben werden. Der letzte Befehl wirkt aber nicht, wenn eine TK Begrenzung überschritten wurde.

Falls es der Admin nicht abgestellt hat, kann sich jeder Spieler durch die Chateingabe: `score` den aktuellen Spielstand anzeigen lassen.

In diesem Abschnitt soll eine einfache Einführung und einige Hilfen gegeben werden, die bei der Installation und Konfiguration dieses Plugins unterstützen sollen.

A.2.1. Voraussetzungen

Es wird ein lauffähiges LogD vorausgesetzt. Darüber hinaus sind noch Einstellungen bei Server und bei Admin Mod durchzuführen.

Server (server.cfg):
```
mp_logdetail "3"
mp_tkpunish "0"
```

Admin Mod (adminmod.cfg):
```
file_access_read 1
file_access_write 1
allow_client_exec 1
amv_enable_beta "menu1"
```

Auf dem Server selbst muss ein Verzeichnis mit dem Namen ffmon im cstrike Verzeichnis erstellt werden, also `cstrike/ffmon`.

Will man die „Chicken"-Option nutzen, ist das Chicken-Metamodplugin[1] zu installieren.

A.2.2. Einstellungen

Zum Verändern der Einstellungen des FFMons benötigt der Admin den Rechtelevel Ban (256). Das Verändern der Einstellungen sollte mit Vorsicht vorgenommen werden, da es mitunter dazu kommen kann, dass auch andere Variablen verändert werden. Die Eingaben erfolgen alle in der Console.

Oft genutzte Einstellungen

Zum Einschalten der TK Überwachung nutzt man die Konsoleneingabe:
```
admin_ffmon TK on
```

Ausschalten der selbigen erfolgt mit:
```
admin_ffmon TK off
```

Um das Chicken in die Bestrafungsliste aufzunehmen gibt man ein:
```
admin_ffmon allowpunish +chicken
```

Das Herausnehmen selbiger erfolgt mit:
```
admin_ffmon allowpunish -chicken
```

Allgemein zugängliche Befehle

`admin_ffmon status` - Anzeige der Einstellungen für alle Variablen des FFMons

[1] http://djeyl.net/chickenmod.php?lang=en

`admin_ffmon showtk` - Anzeige aller TKs für alle Spieler

`admin_ffmon showtd` - Anzeige aller TDs für alle Spieler

TK Einstellungen

`admin_ffmon TK <on/off>` - TK Bestrafung ein/aus

`admin_ffmon tklimit <Wert>` (Default: 3) - Limit unvergebener TKs ab dem gebannt wird

`admin_ffmon tkbantime <Wert>` (Default: 30) - Bannzeit in Minuten

`admin_ffmon tkmenu <on/off>` (Default: off) - Das TKMenu wird ein-/ausgeschaltet

`admin_ffmon tkreset <on/off>` (Default: off) - Setzt das TK Konto der Spieler bei Mapende auf 0

`admin_ffmon tksave <on/off>` (Default: on) - Bestimmt, ob TKs über einen längeren Zeitraum gespeichert werden

`admin_ffmon tksavetime <Wert>` (Default: 24) - Zeitraum des Speicherns von TKs in Stunden

`admin_ffmon tksavelimit <Wert>` (Default: 6) - Limit unvergebener TKs innerhalb der Savetime

`admin_ffmon tksavebantime <Wert>` (Default: 1440) - Bannzeit in Minuten bei Überschreitung des TKSavelimits

`admin_ffmon tksavepurgetime <Wert>` (Default: 24) - Zeitraum in Stunden, in denen die TK Dateien bereinigt werden

TD Einstellungen

`admin_ffmon td on/off` - TD Bestrafung ein/aus

`admin_ffmon tdlimit <Wert>` (Default: 5) - Limit bei dem eine Bestrafung erfolgt

`admin_ffmon tdaction slay/kick/ban/bury/chicken` (Default: slay) - Bestrafungsart einstellen

Einstellungen zum Rundenstart

`admin_ffmon rs <Wert>` (Default: 10) - Rundenstartzeit in Sekunden mit verschärfter Bestrafung

`admin_ffmon rsbantk on/off` (Default: on) - Bannen in der Rundenstartzeit ein/ausschalten

`admin_ffmon rsbantktime <Wert>` (Default: 300) - Bannzeit in Minuten bei TK in der Rundenstartzeit

`admin_ffmon rstdslap on/off` (Default: on) - TD Bestrafung Slap ein/ausschalten

`admin_ffmon rstdslapcount <Wert>` (Default: 5) - Anzahl der Slaps bei TD in der Rundenanfangszeit

`admin_ffmon rstdaction none/slay/kick/ban/chicken` (Default: ban) - Art der Bestrafung bei Überschreitung des TD Limits in der Rundenstartzeit

`admin_ffmon rstdcount <Wert>` (Default: 3) - Festlegen des TD Limits während der Rundenstartzeit

`admin_ffmon rstdbantime <Wert>` (Default: 2) - Bannzeit in Minuten bei Erreichen des TD Limits in der Rundenstartzeit

`admin_ffmon ss on/off` (Default: off) - ein/ausschalten der Sofortschussregistrierung

`admin_ffmon sstime <Wert>` (Default: 2) - Dauer in Sekunden der Sofortschussregistrierung am Rundenanfang

`admin_ffmon ssaction none/slay/kick/ban/bury` (Default: kick) - Bestrafung bei Sofortschüssen am Rundenanfang

`admin_ffmon ssbantime <Wert>` (Default: 5) - Bannzeit in Minuten bei Sofortschüssen und der Bestrafung Bann

Einstellungen zum Teambalancing

`admin_ffmon skunk on/off` (Default: on) - Überwachung der Teams bezüglich eines Winlimits

`admin_ffmon skunktype 0/1` (Default: 0) - normale oder traditionelle Winlimitüberwachung (traditionell = das Verliererteam hat keine Runde gewonnen)

`admin_ffmon skunklimit <Wert>` (Default: 10) - Unterschied gewonnener Runden zwischen den Teams, damit ein Mapwechsel vollzogen wird

Einstellungen zur Admin-Immunität

`admin_ffmon immunity on/off` (Default: off) - Macht Admins immun gegen Bestrafungen des Plugins

`admin_ffmon immunitylevel <Wert>` (Default: 65536) - Rechtelevel den immune Admins haben müssen

Abstimmungseinstellungen

`admin_ffmon startvote on/off` (Default: on) - Startet einen HLDS Vote, bevor die in mp_winlimit angegebene Rundenzahl erreicht ist

`admin_ffmon startvoteround <Wert>` (Default: 2) - Rundenzahl bevor mp_winlimit erreicht ist, um einen HLDS Vote zu starten

Sonstige Einstellungen

`admin_ffmon <on/off>` - Schaltet den FFMon ein/aus (die Eingabe ohne Wert zeigt den Status)

`admin_ffmon scores on/off` (Default: on) - Zeigt Teamscores am Ende jeder Runde

`admin_ffmon log on/off` (Default: on) - Ermöglicht FFMon Einträge in die Logdateien

`admin_ffmon consgreet on/off` (Default: on) - Konsolenbegrüßung (seit CS 1.6 nicht mehr relevant)

`admin_ffmon blockattack on/off` (Default: on) - Unterdrückt Einträge von Beschädigungen in die Logdateien. Entlastet den Server. Wer allerdings ausführliche Stats wünscht, schaltet blockattack ab

`admin_ffmon allowpunish kick/ban/slap/bury/chicken/overrideban/all` (Default: alles außer chicken) - Bestrafungsart bei TKs

`admin_ffmon userslapcount <Wert>` (Default: 1) - Zahl der Schläge, wenn Slap als TK Bestrafung genutzt wird

`admin_ffmon overridelevel <Wert>` (Default: 0) - erforderlicher Rechtelevel um einen Bann aufzuheben

`admin_ffmon buryglowtime <Wert>` (Default: 30) - Leuchtzeit in Sekunden bei der TK Bestrafung bury

`admin_ffmon_purge` - Löst eine Bereinigung der TK Dateien aus

`admin_ffmon_reload` - Lädt die Einstellungen aus der vault.ini neu, z.B. wenn diese manuell editiert wurde

Tipps für die Praxis

Für Funmaps, z.B. fy_iceworld, sind die Einstellungen des FFMon meistens zu streng. Man kann die Einstellungen jedoch für spezielle Maps entschärfen, indem man mit mapspezifischen Konfigurationsdateien arbeitet. Eingetragen wird dort z.B. Folgendes

```
admin_cmd admin_ffmon rs 4
```

Damit wird der Wert der Rundenstartzeit für diese Map auf 4 Sekunden herabgesetzt.

Da dieser Wert auf allen Maps erhalten bleibt, muss man einen Eintrag in die server.cfg eintragen (hinter `exec addons/adminmod/config/adminmod.cfg`), um die Grundeinstellung beim Mapwechsel zurückzuerlangen:

```
admin_cmd admin_ffmon rs 10
```

Dazu muss gewährleistet werden, dass die server.cfg bei jedem Mapwechsel ausgeführt wird. Dies erreicht man mit dem folgenden Eintrag:

```
mapchangecfgfile server.cfg
```

A.2.3. Probleme

FFMon funktioniert aber nicht

Es ist mittels Consoleneingabe zu Überprüfen, ob LogD korrekt installiert ist:

```
rcon meta list
```

Die Rückgabe müsste dann Folgendes enthalten:

```
[x] LogDaemon RUN - logd_mm_i386.so v1.0.0.6 ini ANY Pause
```

Des Weiteren ist zu überprüfen, ob `mp_logdetail 3` in der server.cfg eingestellt ist.

TKs werden nicht über 24 Stunden behalten

Es ist zu überprüfen, ob file_access_read und file_access_write in der adminmod.cfg jeweils auf 1 eingestellt ist? Anderenfalls kann das Plugin die TKs nicht lesen und/oder speichern.

Beim TK-Opfer erscheint kein Menu

Es ist in die adminmod.cfg zu überprüfen, ob der Eintrag `amv_enable_beta "menu1"` eingetragen ist. Weiterhin sollte überprüft werden, ob `admin_ffmon tkmenu on` eingestellt ist.

Trotz `forgive` wird der Täter am Rundenanfang getötet

Dieser Effekt kann auftreten, wenn in der server.cfg die Variable `mp_tkpunish` nicht auf 0 gesetzt wurde.

Quelle: Stillsetzhut

B. Sonstiges

B.1. Was macht eigentlich Setinfo?

Jeder kennt sicherlich Servervariablen (Beispiel: mp_timelimit). Diese Variablen sind fest in der Serversoftware verdrahtet. Man kann nur deren Wert verändern, z.B. in der server.cfg oder als Rcon-Befehl. Analog dazu gibt es auch festverdrahtete Clientvariablen (Beispiel: rate). Auch hier kann nur der Wert verändert werden (config.cfg oder Client-Console).

Jetzt nehmen wir an, dass eine Serverapplikation wie Admin Mod beim Client eine Variable auslesen möchte. Es gibt natürlich keine Möglichkeit Half-Life neu zu compilieren mangels Quellcode. Valve hat daher benutzerdefinierte Variablen eingeführt. Diese definiert man mittels des setinfo Befehls.

Anbei ein Beispiel für Sir Drink a lot's Plugins. Über ein benutzerdefinierte Clientvariable liest es aus, ob der Client Customsounds hören will:

```
setinfo "am_sound" "0"
```

Man definiert hier eine Variable am_sound mit dem Wert 0. Es ist also nichts anderes als eine Variablendefinition unter SMALL (new am_sound=0). Wenn man diesen Befehl in der config.cfg verankert, wird man auf keinem Server mit Sir Drink a lot's Plugins mehr von den zugehörigen Sounds genervt.

Woher weiß das Plugin aber, was der Client eingetragen hat? Dafür gibt es die Scripting-Funktion get_userinfo(). Damit kann man einige Clientvariablen auslesen bzw. alle mit setinfo definierten Variablen.

Man könnte jetzt auf die Idee kommen, jedes Plugin mit setinfo-Zeilen auszustatten (z.B. Plugin für Client an oder aus). Schließlich kann man diese Variablen einfach per execclient() Befehl beim Client in die config.cfg schreiben (soweit er sie nicht schreibgeschützt hat; das empfehle ich jedem, da es inzwischen offensichtlich Hobby geworden ist fremde Configs zu zerstören). Leider ist der Speicherbereich, der setinfo-Variablen zur Verfügung gestellt werden, sehr klein. Wer zuviele oder zu lange Variablen hat, bekommt oft ein „Info String Exceeded“ in seiner Console, meist verbunden mit einem „Name reserved“ und einem fluchenden Spieler, der Admin Mod dafür verantwortlich macht. Meist ist aber Statsme dafür verantwortlich, was meist 2 lange setinfos in der config.cfg hinterließ. Man mag soviel über die Additionsorgie unter Adminmod meckern, aber den Autoren war dieses Problem durchaus bewusst. Durch dieses System wird massiv Speicher gespart! Will man also setinfo in seinem Plugin verwenden, so sollte man

dies nur tun, wenn man es als allgemeine Einstellungen für fast alle Plugins nutzen kann. Vielleicht nutzen ja mehr Plugin-Programmierer am_sound. Zu hoffen wäre es.

B.2. Interaktion zwischen HL Engine und GameDLL

Die HL Engine lädt beim Starten dynamisch eine GameDLL. Welche das ist, wird aus den Kommandozeilen Parameter -game und der liblist.gam Datei in dem angegebenen Verzeichnis ermittelt. Diese GameDLL ist verantwortlich für die Elemente des Spiels.

`Engine ↔ GameDLL`

Wenn in der liblist.gam die Metamod DLL angegeben ist, lädt die Engine die Metamod DLL. Der Engine ist es egal, was sie für eine DLL lädt, solange sie alle Funktionen bereithält, um erfolgreich geladen zu werden.

Da Metamod aber kein Spiel ist, muss irgendwo noch die GameDLL herkommen. Metamod schaut also nach, was die Engine für ein Spiel laden wollte, überprüft in seiner internen Liste, welche GameDLL dazu passt und lädt dann die GameDLL. Die Engine denkt sie unterhält sich mit der GameDLL, die GameDLL denkt sie unterhält sich mit der Engine, Metamod sitzt dazwischen und alle sind glücklich.

`Engine ↔ Metamod ↔ GameDLL`

Alte Bots, die keine Metamod Plugins sind, verhalten sich exakt so, wie es auch Metamod tut.

`Engine ↔ BotDLL ↔ GameDLL`

Wenn man also Metamod und Bots gleichzeitig laden will, dann müssen sie hintereinander geschaltet werden, denn nach oben (links) denkt jeder er sieht die Engine und nach unten (rechts) denkt jeder er sieht die GameDLL. Wenn Metamod also eine BotDLL lädt, dann tritt diese für Metamod als GameDLL auf, genauso wie das die BotDLL für die Engine tun würde. Man kann Metamod anweisen, eine andere GameDLL zu benutzen als die, die er normalerweise benutzen würde. In unserem Falle also eine BotDLL. Die Engine lädt Metamod als GameDLL. Metamod ist keine GameDLL und lädt daher die BotDLL als GameDLL. Die BotDLL ist auch keine GameDLL und lädt daher die GameDLL, in welcher jetzt tatsächlich das Spiel steckt.

`Engine ↔ Metamod ↔ BotDLL ↔ GameDLL`

Es geht auch andersherum:

`Engine ↔ BotDLL ↔ Metamod ↔ GameDLL`

Das hat aber den Nachteil, dass Metamod, und damit seine Plugins, keinen Einfluss mehr auf die BotDLL haben.

Frage: Wie soll jetzt Metamod eine zweite GameDLL laden, wenn a) nur eine GameDLL existieren kann (die BotDLL sieht für Metamod aus wie eine GameDLL) und b) diese

in die Reihe passen muss, aber Metamod nur die Position rechts von sich beeinflussen kann?

Faktenlage B:
Um Metamod mitzuteilen, dass es eine andere GameDLL laden soll, als die, die es normalerweise aussuchen würde, setzt man ein Schlüssel-Wert Paar, die in der Liste namens „localinfo“. Als Schlüssel nimmt man dafür „mm_gamedll“ und der Wert enthält den Pfad zur GameDLL z.B. „podbot/podbot.dll“. Danach hat man folgende Liste:

```
(mm_gamedll, podbot/podbot.dll)
```

Metamod schaut nach, ob es einen Eintrag unter dem Schlüssel „mm_gamedll“ gibt, und verwendet diesen um die dort angegebene „Game“DLL zu laden. Die Kommandozeile

```
+localinfo mm_gamedll podbot/podbot.dll
```

legt dieses Paar unter diesem Schlüssel mit diesem Wert in der Engine an. Was passiert, wenn dieser Schlüssel zweimal mit einem Wert belegt wird?

```
+localinfo mm_gamedll podbot/podbot.dll +localinfo mm_gamedll
dummbot/dumm.dll
```

Da ein Schlüssel in nur genau einem Schlüssel-Wert Paar vorkommen kann (sonst wäre er ja nicht eindeutig), resultiert daraus logischerweise folgende Liste:

```
(mm_gamedll, dummbot/dumm.dll)
```

Die erste Paarbelegung wird durch ein erneutes Zuweisen überschrieben. Der letzte Eintrag gilt. Genauso wie, wenn in der server.cfg die selbe Cvar mehrfach belegt wird; es gilt nur die letzte Zuweisung.

Schlussfolgerung: Es ist technisch und logisch nicht möglich zwei GameDLL zu laden und es ist ebenso nicht möglich, zwei Paare unter demselben Schlüssel anzulegen.

Autor: Da Rope

B.3. Res-Dateien

Resourcendateien oder kurz RES-Dateien sind Textdateien, welche sich im Maps Verzeichnis des Mods befinden. Der Dateiname entspricht dem Mapnamen, wie zum Beispiel:

```
de_dust.res
```

Man benötigt derartige Dateien um Sounds, Texte oder Teile von Maps vom Server zum Client zu übertragen.

Wie bereits erwähnt sind RES-Dateien einfache Textdateien und können somit über jeden Editor wie beispielsweise Notepad erstellt oder verändert werden.

Da die RES-Dateien zum Übertragen von Dateien vom Server zum Client benutzt werden, müssen die zu übertragenden Dateien auf dem Server vorhanden sein. In der RES-

Datei wird nun der Dateiname und der zugehörige Pfad relativ zum Mod Pfad angegeben.

Beispiel:
Angenommen es befindet sich eine Datei namens „woohoo.wav“ im Verzeichnis ...\cstrike\sound\misc\ des Servers. Diese soll zum Client übertragen, wenn zur Map de_dust gewechselt wird. Dazu muss eine Datei namens de_dust.res im Ordner maps des Servers erstellt werden und die folgende Zeile hineingeschrieben werden:

```
sound/misc/woohoo.wav
```

Mit jedem Mapwechsel wird die korrespondierende RES-Datei verarbeitet. Dazu muss sich die zugehörige Datei im Verzeichnis befinden, in dem sich die Map-Datei befindet.

Beispiel:
Wird zur Map de_dust gewechselt, wird nach einer Datei namens de_dust.res im Maps-Verzeichnis gesucht und diese dann (sofern vorhanden) verarbeitet.

Man muss nicht zu jeder Map im Maps-Ordner des Servers eine korrespondierende RES-Datei erstellen. Es bietet sich jedoch an, da einem Player der gerade den Server betritt in jedem Fall die Dateien übertragen werden.

Autor: Biohazard

Das Customplugin plugin_bk_res[1] kann bei der Erstellung der Dateien helfen.

B.4. HL-Sounds

Einige Sounds sind bereits in Half-Life vorhanden. Sie sind nicht in echten Verzeichnissen vorhanden, sondern werden aus den sogenannten PAK-Dateien geladen. Will man beispielsweise Barney sprechen lassen, kann man in der Serverconsole Folgendes eingeben:

```
speak barney/somethingstinky
```

Oder will man eine Frau sprechen lassen, dass noch 15 Sekunden verbleiben:

```
speak fvox/fifteen seconds remaining
```

Die Verwendung mit Admin Mod kann mit dem Customplugin plugin_speech[2] vereinfacht werden.

Alle folgenden Sounds sind bereits beim Spieler vorhanden, und müssen nicht heruntergeladen werden (Thema: Res-Dateien). Sie wurden von Kleen13 zusammengetragen[3].

[1] http://www.adminmod.de/plugins.php?plugin=plugin_bk_res
[2] http://www.adminmod.de/plugins.php?plugin=plugin_speech
[3] kleen13@hotmail.com (MSN), 5402104 (ICQ)

B.4.1. agrunt

ag_alert1.wav	ag_die2.wav	ag_idle4.wav
ag_alert2.wav	ag_die3.wav	ag_idle5.wav
ag_alert3.wav	ag_die5.wav	ag_pain1.wav
ag_alert4.wav	ag_fire1.wav	ag_pain2.wav
ag_alert5.wav	ag_fire2.wav	ag_pain3.wav
ag_attack1.wav	ag_fire3.wav	ag_pain4.wav
ag_attack2.wav	ag_idle1.wav	ag_pain5.wav
ag_attack3.wav	ag_idle2.wav	
ag_die1.wav	ag_idle3.wav	

B.4.2. ambience

_comma.wav	iotone.wav	labdrone1.wav
alien_beacon.wav	boomer.wav	labdrone2.wav
alien_blipper.wav	reather.wav	labgear.wav
alien_builder.wav	burning1.wav	labmoan.wav
alien_chatter.wav	burning2.wav	littlemachine.wav
alien_creeper.wav	burning3.wav	loader_hydra1.wav
alien_cycletone.wav	computalk1.wav	loader_step1.wav
alien_frantic.wav	cricket.wav	mechwhine.wav
alien_hollow.wav	rickets.wav	mgun_burst1.wav
alien_humongo.wav	crtnoise.wav	mgun_burst2.wav
alien_minddrill.wav	deadsignal1.wav	mgun_burst3.wav
alien_powernode.wav	deadsignal2.wav	mgun_burst4.wav
alien_purrmachine.wav	des_wind1.wav	particle_suck1.wav
alien_squit.wav	des_wind2.wav	particle_suck2.wav
alien_twow.wav	des_wind3.wav	port_suckin1.wav
alien_zonerator.wav	disgusting.wav	port_suckout1.wav
aliencave1.wav	distantmortar1.wav	pounder.wav
alienclicker1.wav	distantmortar2.wav	pulsemachine.wav
alienfazzle1.wav	distantmortar3.wav	pumper.wav
lienflyby1.wav	drips.wav	quail1.wav
alienlaser1.wav	dronemachine1.wav	rifle1.wav
alienvalve1.wav	dronemachine2.wav	rifle2.wav
alienvalve2.wav	dronemachine3.wav	rocket_groan1.wav
alienvoices1.wav	flameburst1.wav	rocket_groan2.wav
alienwind1.wav	flies.wav	rocket_groan3.wav
alienwind2.wav	hammer.wav	rocket_groan4.wav
bee1.wav	hawk1.wav	rocket_steam1.wav
bee2.wav	industrial1.wav	rocketflame1.wav
biggun1.wav	industrial2.wav	rocketrumble1.wav
biggun2.wav	industrial3.wav	rotormachine.wav
biggun3.wav	industrial4.wav	sandfall1.wav
bigwarning.wav	jetflyby1.wav	sandfall2.wav

screammachine.wav	tankdrivein2.wav	warn2.wav
signalgear1.wav	tankidle1.wav	warn3.wav
signalgear2.wav	tankidle2.wav	waterfall1.wav
siren.wav	techamb2.wav	waterfall2.wav
sparks.wav	the_horror1.wav	waterfall3.wav
squeeks1.wav	the_horror2.wav	wind1.wav
squeeks2.wav	the_horror3.wav	wind2.wav
squirm2.wav	the_horror4.wav	wren1.wav
squitch.wav	truck1.wav	xtal_down1.wav
squitch2.wav	truck2.wav	zapmachine.wav
steamburst1.wav	turretrot1.wav	zipmachine.wav
steamjet1.wav	turretrot2.wav	
tankdrivein1.wav	warn1.wav	

B.4.3. apache

ap_rotor1.wav	ap_rotor3.wav	ap_whine1.wav
ap_rotor2.wav	ap_rotor4.wav	

B.4.4. aslave

slv_alert1.wav	slv_pain1.wav	slv_word5.wav
slv_alert3.wav	slv_word1.wav	slv_word6.wav
slv_alert4.wav	slv_word2.wav	slv_word7.wav
slv_die1.wav	slv_word3.wav	slv_word8.wav
slv_die2.wav	slv_word4.wav	

B.4.5. barnacle

bcl_alert2.wav	bcl_chew2.wav	bcl_die3.wav
bcl_bite3.wav	bcl_chew3.wav	bcl_tongue1.wav
bcl_chew1.wav	bcl_die1.wav	

B.4.6. barney

_comma.wav	ba_attack2.wav	ba_dotoyou.wav
aimforhead.wav	ba_bring.wav	ba_duty.wav
aintgoin.wav	ba_buttugly.wav	ba_endline.wav
aintscared.wav	ba_close.wav	ba_firepl.wav
alreadyasked.wav	ba_die1.wav	ba_friends.wav
ambush.wav	ba_die2.wav	ba_gotone.wav
ba_another.wav	ba_die3.wav	ba_iwish.wav
ba_attack1.wav	ba_dontmake.wav	ba_later.wav

ba_pain1.wav	c2a3_ba_assn.wav	hearsomething2.wav
ba_pain2.wav	c2a4_ba_1tau.wav	hellonicesuit.wav
ba_pain3.wav	c2a4_ba_3tau.wav	helpothers.wav
ba_pissme.wav	c2a4_ba_5tau.wav	heybuddy.wav
ba_post.wav	c2a4_ba_alive.wav	heyfella.wav
ba_raincheck.wav	c2a4_ba_arg1a.wav	hitbad.wav
ba_seethat.wav	c2a4_ba_arg3a.wav	howdy.wav
ba_somuch.wav	c2a4_ba_arg5a.wav	howyoudoing.wav
ba_stepoff.wav	c2a4_ba_longnite.wav	icanhear.wav
ba_tomb.wav	c2a4_ba_steril.wav	iguess.wav
ba_uwish.wav	c2a4_ba_teach.wav	illwait.wav
ba_watchit.wav	c2a5_ba_helpme.wav	imdead.wav
ba_whatyou.wav	c2a5_ba_letout.wav	imhit.wav
ba_whoathere.wav	c2a5_ba_rpg.wav	imwithyou.wav
badarea.wav	c2a5_ba_sniped.wav	ireckon.wav
badfeeling.wav	c3a1_ba_1sat.wav	iwaithere.wav
beertopside.wav	c3a1_ba_3sat.wav	justdontknow.wav
bequiet.wav	c3a1_ba_5sat.wav	leavealone.wav
bigmess.wav	c3a2_ba_2surv.wav	letsgo.wav
bigplace.wav	c3a2_ba_4surv.wav	letsmoveit.wav
c1a0_ba_button.wav	c3a2_ba_6surv.wav	luckwillturn.wav
c1a0_ba_desk.wav	c3a2_ba_8surv.wav	maybe.wav
c1a0_ba_headdown.wav	c3a2_ba_stay.wav	missingleg.wav
c1a0_ba_hevno.wav	cantfigure.wav	mrfreeman.wav
c1a0_ba_hevyes.wav	checkwounds.wav	nodrill.wav
c1a0_ba_late.wav	coldone.wav	nope.wav
c1a1_ba_glad.wav	crewdied.wav	nosir.wav
c1a2_ba_2zomb.wav	diebloodsucker.wav	notelling.wav
c1a2_ba_4zomb.wav	dobettertogether.wav	noway.wav
c1a2_ba_bullsquid.wav	dontaskme.wav	openfire.wav
c1a2_ba_climb.wav	dontbet.wav	realbadwound.wav
c1a2_ba_goforit.wav	dontbuyit.wav	rightway.wav
c1a2_ba_slew.wav	dontfigure.wav	seeya.wav
c1a2_ba_surface.wav	dontguess.wav	slowingyoudown.wav
c1a2_ba_top.wav	donthurtem.wav	somethingdied.wav
c1a4_ba_octo1.wav	dontreckon.wav	somethingmoves.wav
c1a4_ba_octo2.wav	getanyworse.wav	somethingstinky.wav
c1a4_ba_octo3.wav	gettingcloser.wav	soundsbad.wav
c1a4_ba_octo4.wav	gladof38.wav	soundsright.wav
c1a4_ba_wisp.wav	gladtolendhand.wav	standback.wav
c2a_ba_hub1a.wav	guyresponsible.wav	standguard.wav
c2a1_ba_again.wav	hardtosay.wav	stench.wav
c2a1_ba_hub1a.wav	haybuddy.wav	stop1.wav
c2a1_ba_hub2a.wav	hayfella.wav	stop2.wav
c2a2_ba_launch.wav	hearsomething.wav	stophere.wav

survive.wav	whatgood.wav	yougotit.wav
targetpractice.wav	whatisthat.wav	youhearthat.wav
teamup1.wav	whatsgoingon.wav	youneedmedic.wav
teamup2.wav	workingonstuff.wav	youtalkmuch.wav
thinking.wav	yessir.wav	yup.wav
waitin.wav	youbet.wav	
wayout.wav	youeverseen.wav	

B.4.7. boid

boid_alert1.wav	boid_idle1.wav	boid_idle3.wav
boid_alert2.wav	boid_idle2.wav	

B.4.8. bullchicken

boid_alert1.wav	bc_attackgrowl3.wav	bc_idle4.wav
boid_alert2.wav	bc_bite1.wav	bc_idle5.wav
boid_idle1.wav	bc_bite2.wav	bc_pain1.wav
boid_idle2.wav	bc_bite3.wav	bc_pain2.wav
bc_acid2.wav	bc_die1.wav	bc_pain3.wav
bc_attack1.wav	bc_die2.wav	bc_pain4.wav
bc_attack2.wav	bc_die3.wav	bc_spithit1.wav
bc_attack3.wav	bc_idle1.wav	bc_spithit2.wav
bc_attackgrowl.wav	bc_idle2.wav	bc_spithit3.wav
bc_attackgrowl2.wav	bc_idle3.wav	

B.4.9. buttons

bell1.wav	button6.wav	lever3.wav
blip1.wav	button7.wav	lever4.wav
blip2.wav	button8.wav	lever5.wav
button1.wav	button9.wav	lightswitch2.wav
button10.wav	latchlocked1.wav	spark1.wav
button11.wav	latchlocked2.wav	spark2.wav
button2.wav	latchunlocked1.wav	spark3.wav
button3.wav	latchunlocked2.wav	spark4.wav
button4.wav	lever1.wav	spark5.wav
button5.wav	lever2.wav	spark6.wav

B.4.10. common

bodydrop1.wav	launch_select1.wav	npc_step4.wav
bodydrop2.wav	launch_select2.wav	null.wav
bodydrop3.wav	launch_upmenu1.wav	wpn_denyselect.wav
bodydrop4.wav	menu1.wav	wpn_hudoff.wav
bodysplat.wav	menu2.wav	wpn_hudon.wav
launch_deny1.wav	menu3.wav	wpn_moveselect.wav
launch_deny2.wav	npc_step1.wav	wpn_select.wav
launch_dnmenu1.wav	npc_step2.wav	
launch_glow1.wav	npc_step3.wav	

B.4.11. controller

con_alert1.wav	con_die1.wav	con_idle5.wav
con_alert2.wav	con_die2.wav	con_pain1.wav
con_alert3.wav	con_idle1.wav	con_pain2.wav
con_attack1.wav	con_idle2.wav	con_pain3.wav
con_attack2.wav	con_idle3.wav	
con_attack3.wav	con_idle4.wav	

B.4.12. debris

beamstart1.wav	bustflesh2.wav	metal3.wav
beamstart10.wav	bustglass1.wav	metal4.wav
beamstart11.wav	bustglass2.wav	metal5.wav
beamstart14.wav	bustglass3.wav	metal6.wav
beamstart15.wav	bustmetal1.wav	pushbox1.wav
beamstart2.wav	bustmetal2.wav	pushbox2.wav
beamstart3.wav	concrete1.wav	pushbox3.wav
beamstart4.wav	concrete2.wav	wood1.wav
beamstart5.wav	concrete3.wav	wood2.wav
beamstart6.wav	flesh1.wav	wood3.wav
beamstart7.wav	flesh3.wav	wood4.wav
beamstart8.wav	flesh5.wav	zap1.wav
beamstart9.wav	flesh6.wav	zap2.wav
bustceiling.wav	flesh7.wav	zap3.wav
bustconcrete1.wav	glass1.wav	zap5.wav
bustconcrete2.wav	glass2.wav	zap6.wav
bustcrate1.wav	glass3.wav	zap7.wav
bustcrate2.wav	glass4.wav	zap8.wav
bustcrate3.wav	metal1.wav	
bustflesh1.wav	metal2.wav	

B.4.13. doors

aliendoor1.wav	doormove3.wav	doorstop2.wav
aliendoor2.wav	doormove4.wav	doorstop3.wav
aliendoor3.wav	doormove5.wav	doorstop4.wav
aliendoor4.wav	doormove6.wav	doorstop5.wav
aliendoor5.wav	doormove7.wav	doorstop6.wav
doormove1.wav	doormove8.wav	doorstop7.wav
doormove10.wav	doormove9.wav	doorstop8.wav
doormove2.wav	doorstop1.wav	

B.4.14. fans

fan1.wav	fan2on.wav	fan4off.wav
fan1off.wav	fan3.wav	fan4on.wav
fan1on.wav	fan3off.wav	fan5.wav
fan2.wav	fan3on.wav	fan5off.wav
fan2off.wav	fan4.wav	fan5on.wav

B.4.15. fvox

_comma.wav	blood_loss.wav	fuzz.wav
_period.wav	blood_plasma.wav	get_44ammo.wav
acquired.wav	blood_toxins.wav	get_44pistol.wav
activated.wav	boop.wav	get_9mmclip.wav
administering_medical.wav	buzz.wav	get_alien_wpn.wav
adrenaline_shot.wav	chemical_detected.wav	get_assault.wav
alert.wav	communications_on.wav	get_assaultgren.wav
am.wav	danger.wav	get_battery.wav
ammo_depleted.wav	deactivated.wav	get_bolts.wav
ammo_pickup.wav	east.wav	get_buckshot.wav
antidote_shot.wav	eight.wav	get_crossbow.wav
antitoxin_shot.wav	eighteen.wav	get_egon.wav
armor_compromised.wav	eighty.wav	get_egonpower.wav
armor_gone.wav	eleven.wav	get_gauss.wav
atmospherics_on.wav	evacuate_area.wav	get_grenade.wav
automedic_on.wav	fifteen.wav	get_medkit.wav
beep.wav	fifty.wav	get_pistol.wav
bell.wav	five.wav	get_rpg.wav
bio_reading.wav	flatline.wav	get_rpgammo.wav
biohazard_detected.wav	four.wav	get_satchel.wav
bleeding_stopped.wav	fourteen.wav	get_shotgun.wav
blip.wav	fourty.wav	get_tripmine.wav

health_critical.wav	ninety.wav	sixteen.wav
health_dropping.wav	north.wav	sixty.wav
health_dropping2.wav	one.wav	south.wav
heat_damage.wav	onehundred.wav	targetting_system.wav
hev_critical_fail.wav	online.wav	ten.wav
hev_damage.wav	pain_block.wav	thirteen.wav
hev_general_fail.wav	percent.wav	thirty.wav
hev_logon.wav	pm.wav	three.wav
hev_shutdown.wav	position.wav	time_is_now.wav
hiss.wav	power_below.wav	time_remaining.wav
hours.wav	power_level_is.wav	torniquette_applied.wav
immediately.wav	power_restored.wav	twelve.wav
innsuficient_medical.wav	powerarmor_on.wav	twenty.wav
internal_bleeding.wav	powermove_on.wav	twentyfive.wav
major_fracture.wav	powermove_overload.wav	two.wav
major_lacerations.wav	radiation_detected.wav	vitalsigns_on.wav
medical_repaired.wav	range.wav	voice_off.wav
meters.wav	remaining.wav	voice_on.wav
minor_fracture.wav	safe_day.wav	warning.wav
minor_lacerations.wav	seconds.wav	weapon_pickup.wav
minutes.wav	seek_medic.wav	weaponselect_on.wav
morphine_shot.wav	seven.wav	west.wav
munitionview_on.wav	seventeen.wav	wound_sterilized.wav
near_death.wav	seventy.wav	your.wav
nine.wav	shock_damage.wav	
nineteen.wav	six.wav	

B.4.16. garg

gar_alert1.wav	gar_die1.wav	gar_idle5.wav
gar_alert2.wav	gar_die2.wav	gar_pain1.wav
gar_alert3.wav	gar_flameoff1.wav	gar_pain2.wav
gar_attack1.wav	gar_flameon1.wav	gar_pain3.wav
gar_attack2.wav	gar_flamerun1.wav	gar_step1.wav
gar_attack3.wav	gar_idle1.wav	gar_step2.wav
gar_breathe1.wav	gar_idle2.wav	gar_stomp1.wav
gar_breathe2.wav	gar_idle3.wav	
gar_breathe3.wav	gar_idle4.wav	

B.4.17. gman

gman_choose1.wav	gman_mumble5.wav	gman_offer.wav
gman_choose2.wav	gman_mumble6.wav	gman_otherwise.wav
gman_mumble1.wav	gman_nasty.wav	gman_potential.wav
gman_mumble2.wav	gman_noreg.wav	gman_stepin.wav
gman_mumble3.wav	gman_noregret.wav	gman_suit.wav
gman_mumble4.wav	gman_nowork.wav	gman_wise.wav

B.4.18. gonarch

gon_alert1.wav	gon_birth2.wav	gon_pain4.wav
gon_alert2.wav	gon_birth3.wav	gon_pain5.wav
gon_alert3.wav	gon_childdie1.wav	gon_sack1.wav
gon_attack1.wav	gon_childdie2.wav	gon_sack3.wav
gon_attack2.wav	gon_childdie3.wav	gon_step1.wav
gon_attack3.wav	gon_die1.wav	gon_step2.wav
gon_birth1.wav	gon_pain2.wav	gon_step3.wav

B.4.19. hassault

hw_gun4.wav	hw_shoot3.wav	hw_spinup.wav
hw_shoot1.wav	hw_spin.wav	
hw_shoot2.wav	hw_spindown.wav	

B.4.20. headcrab

hc_alert1.wav	hc_die2.wav	hc_idle5.wav
hc_alert2.wav	hc_headbite.wav	hc_pain1.wav
hc_attack1.wav	hc_idle1.wav	hc_pain2.wav
hc_attack2.wav	hc_idle2.wav	hc_pain3.wav
hc_attack3.wav	hc_idle3.wav	
hc_die1.wav	hc_idle4.wav	

B.4.21. hgrunt

_comma.wav	alert!.wav	am!.wav
_period.wav	alert.wav	am.wav
a!.wav	alien!.wav	anything!.wav
a.wav	alien.wav	are!.wav
affirmative!.wav	all!.wav	are.wav
affirmative.wav	all.wav	area!.wav

area.wav	control!.wav	got!.wav
ass!.wav	control.wav	got.wav
ass.wav	cover!.wav	gr_alert1.wav
at!.wav	creeps!.wav	gr_die1.wav
away!.wav	creeps.wav	gr_die2.wav
backup!.wav	damn!.wav	gr_die3.wav
backup.wav	damn.wav	gr_idle1.wav
bag!.wav	delta!.wav	gr_idle2.wav
bastard!.wav	delta.wav	gr_idle3.wav
stard.wav	down!.wav	gr_loadtalk.wav
blow!.wav	down.wav	gr_mgun1.wav
bogies!.wav	east!.wav	gr_mgun2.wav
bogies.wav	east.wav	gr_mgun3.wav
bravo!.wav	echo!.wav	gr_pain1.wav
bravo.wav	echo.wav	r_pain2.wav
c2a2_hg_chat1a.wav	eightymeters.wav	gr_pain3.wav
c2a2_hg_chat2a.wav	eliminate.wav	gr_pain4.wav
c2a2_hg_chat3a.wav	everything.wav	Sgr_pain5.wav
c2a2_hg_chat4a.wav	fall!.wav	gr_reload1.wav
c2a2_hg_chat5a.wav	fiftymeters.wav	gr_step1.wav
c2a3_ambush_fx.wav	fight!.wav	gr_step2.wav
c2a3_ambush_vox.wav	fight.wav	gr_step3.wav
c2a3_hg_1drag.wav	fire!.wav	gr_step4.wav
c2a3_hg_2drag.wav	fire.wav	grenade!.wav
c2a3_hg_3drag.wav	five!.wav	guard!.wav
c2a3_hg_4drag.wav	five.wav	guard.wav
c2a3_hg_5drag.wav	fivemeters.wav	have!.wav
c2a3_hg_laugh.wav	force!.wav	have.wav
c2a5_hg_abandon.wav	force.wav	he!.wav
c2a5_hg_lebuz.wav	formation!.wav	heavy!.wav
call!.wav	formation.wav	heavy.wav
casualties!.wav	fortymeters.wav	hell!.wav
charlie!.wav	four!.wav	hell.wav
charlie.wav	four.wav	here!.wav
check!.wav	foxtrot!.wav	here.wav
check.wav	foxtrot.wav	hg_civvies.wav
checking!.wav	freeman!.wav	hg_sucks.wav
checking.wav	freeman.wav	hold!.wav
clear!.wav	get!.wav	hold.wav
clear.wav	go!.wav	hole!.wav
clik.wav	go.wav	hole.wav
command!.wav	god!.wav	hostiles!.wav
command.wav	god.wav	hostiles.wav
continue!.wav	going!.wav	hot!.wav
continue.wav	going.wav	hot.wav

hundredmeters.wav	objective!.wav	shot!.wav
i!.wav	objective.wav	shot.wav
i.wav	of!.wav	sign!.wav
in!.wav	of.wav	sign.wav
in.wav	oh!.wav	signs!.wav
is!.wav	ok!.wav	signs.wav
is.wav	ok.wav	silence!.wav
kick!.wav	one!.wav	silence.wav
lay!.wav	one.wav	sir!.wav
left!.wav	onefiftymeters.wav	sir.wav
left.wav	orders!.wav	six!.wav
lets!.wav	orders.wav	six.wav
lets.wav	our!.wav	sixtymeters.wav
level!.wav	out!.wav	some!.wav
level.wav	out.wav	some.wav
lookout!.wav	over!.wav	something!.wav
lookout.wav	over.wav	something.wav
maintain!.wav	patrol!.wav	south!.wav
maintain.wav	patrol.wav	south.wav
mission!.wav	people!.wav	squad!.wav
mission.wav	people.wav	squad.wav
mister!.wav	position!.wav	stay!.wav
mister.wav	position.wav	stay.wav
mother!.wav	post!.wav	suppressing!.wav
move!.wav	post.wav	sweep!.wav
move.wav	private!.wav	sweep.wav
movement!.wav	private.wav	take!.wav
movement.wav	quiet!.wav	tango!.wav
moves!.wav	quiet.wav	tango.wav
moves.wav	radio!.wav	target!.wav
my!.wav	radio.wav	target.wav
my.wav	recon!.wav	team!.wav
need!.wav	recon.wav	team.wav
negative!.wav	request!.wav	tenmeters.wav
negative.wav	right!.wav	that!.wav
neutralize!.wav	right.wav	that.wav
neutralized!.wav	roger!.wav	the!.wav
niner!.wav	roger.wav	the.wav
niner.wav	sector!.wav	there!.wav
no!.wav	sector.wav	there.wav
no.wav	secure!.wav	these!.wav
north!.wav	secure.wav	these.wav
north.wav	seventymeters.wav	thirtymeters.wav
nothing!.wav	shit!.wav	this!.wav
nothing.wav	shit.wav	this.wav

those!.wav	we!.wav	yes.wav
those.wav	we.wav	yessir!.wav
three!.wav	weapons!.wav	yessir.wav
three.wav	weapons.wav	you!.wav
tight!.wav	weird!.wav	your!.wav
tight.wav	weird.wav	your.wav
twentymeters.wav	west!.wav	zero!.wav
two!.wav	west.wav	zero.wav
two.wav	we've!.wav	zone!.wav
twohundredmeters.wav	we've.wav	zone.wav
uhh.wav	will!.wav	zulu!.wav
under!.wav	yeah!.wav	zulu.wav
up!.wav	yeah.wav	
up.wav	yes!.wav	

B.4.22. holo

tr_ba_lightson.wav	tr_holo_grenade.wav	tr_holo_pullbox.wav
tr_ba_unuse.wav	tr_holo_hitall.wav	tr_holo_pushbox.wav
tr_ba_use.wav	tr_holo_injury.wav	tr_holo_radiation.wav
tr_holo_3jumps.wav	tr_holo_intro.wav	tr_holo_retry.wav
tr_holo_breakbox.wav	tr_holo_jduck.wav	tr_holo_runstart.wav
tr_holo_breath.wav	tr_holo_jump.wav	tr_holo_spinbridge.wav
tr_holo_button.wav	tr_holo_jumpdown.wav	tr_holo_startlift.wav
tr_holo_charger.wav	tr_holo_jumpgap.wav	tr_holo_steam.wav
tr_holo_commencing.wav	tr_holo_keeptrying.wav	tr_holo_target.wav
tr_holo_congrats.wav	tr_holo_ladder.wav	tr_holo_tryagain.wav
tr_holo_done.wav	tr_holo_leadguard.wav	tr_holo_usetrain.wav
tr_holo_drown.wav	tr_holo_lightoff.wav	tr_sci_goodwork.wav
tr_holo_duck.wav	tr_holo_longjump.wav	tr_sci_hardlynoticed.wav
tr_holo_fallshort.wav	tr_holo_medkit.wav	tr_sci_nextstation.wav
tr_holo_fantastic.wav	tr_holo_move.wav	tr_sci_unuse.wav
tr_holo_flashlight.wav	tr_holo_nicejob.wav	tr_sci_use.wav
tr_holo_greatwork.wav	tr_holo_pipeduck.wav	

B.4.23. hornet

ag_buzz1.wav	ag_buzz3.wav	ag_hornethit2.wav
ag_buzz2.wav	ag_hornethit1.wav	ag_hornethit3.wav

B.4.24. houndeye

he__alert1.wav	he__die1.wav	he__idle3.wav
he__alert2.wav	he__die2.wav	he__idle4.wav
he__alert3.wav	he__die3.wav	he__pain1.wav
he__attack1.wav	he__hunt1.wav	he__pain2.wav
he__attack2.wav	he__hunt2.wav	he__pain3.wav
he__attack3.wav	he__hunt3.wav	he__pain4.wav
he__blast1.wav	he__hunt4.wav	he__pain5.wav
he__blast2.wav	he__idle1.wav	
he__blast3.wav	he__idle2.wav	

B.4.25. ichy

ichy__alert1.wav	ichy__die1.wav	ichy__idle4.wav
ichy__alert2.wav	ichy__die2.wav	ichy__pain1.wav
ichy__alert3.wav	ichy__die3.wav	ichy__pain2.wav
ichy__attack1.wav	ichy__die4.wav	ichy__pain3.wav
ichy__attack2.wav	ichy__idle1.wav	ichy__pain5.wav
ichy__bite1.wav	ichy__idle2.wav	
ichy__bite2.wav	ichy__idle3.wav	

B.4.26. items

9mmclip1.wav	guncock1.wav	medshotno1.wav
9mmclip2.wav	gunpickup1.wav	smallmedkit1.wav
airtank1.wav	gunpickup2.wav	smallmedkit2.wav
ammopickup1.wav	gunpickup3.wav	suitcharge1.wav
ammopickup2.wav	gunpickup4.wav	suitchargeno1.wav
clipinsert1.wav	medcharge4.wav	suitchargeok1.wav
cliprelease1.wav	medshot4.wav	weapondrop1.wav
flashlight1.wav	medshot5.wav	

B.4.27. leech

leech__alert1.wav	leech__bite1.wav	leech__bite3.wav
leech__alert2.wav	leech__bite2.wav	

B.4.28. nihilanth

nil__alone.wav	nil__die.wav	nil__last.wav
nil__comes.wav	nil__done.wav	nil__man__notman.wav
nil__deceive.wav	nil__freeman.wav	nil__now__die.wav

nil_slaves.wav	nil_thetruth.wav	nil_win.wav
nil_thelast.wav	nil_thieves.wav	

B.4.29. plats

bigmove1.wav	heavystop1.wav	talkstop1.wav
bigmove2.wav	heavystop2.wav	train_use1.wav
bigstop1.wav	platmove1.wav	train1.wav
bigstop2.wav	platstop1.wav	train2.wav
elevbell1.wav	rackmove1.wav	ttrain_brake1.wav
elevmove1.wav	rackstop1.wav	ttrain_start1.wav
elevmove2.wav	railmove1.wav	ttrain1.wav
elevmove3.wav	railstop1.wav	ttrain2.wav
freightmove1.wav	squeekmove1.wav	ttrain3.wav
freightmove2.wav	squeekstop1.wav	ttrain4.wav
freightstop1.wav	talkmove1.wav	ttrain6.wav
heavymove1.wav	talkmove2.wav	ttrain7.wav

B.4.30. player

breathe1.wav	pl_grate4.wav	pl_slosh3.wav
geiger1.wav	pl_jump1.wav	pl_slosh4.wav
geiger2.wav	pl_jump2.wav	pl_step1.wav
geiger3.wav	pl_jumpland2.wav	pl_step2.wav
geiger4.wav	pl_ladder1.wav	pl_step3.wav
geiger5.wav	pl_ladder2.wav	pl_step4.wav
geiger6.wav	pl_ladder3.wav	pl_swim1.wav
heartbeat1.wav	pl_ladder4.wav	pl_swim2.wav
pl_dirt1.wav	pl_metal1.wav	pl_swim3.wav
pl_dirt2.wav	pl_metal2.wav	pl_swim4.wav
pl_dirt3.wav	pl_metal3.wav	pl_tile1.wav
pl_dirt4.wav	pl_metal4.wav	pl_tile2.wav
pl_duct1.wav	pl_pain2.wav	pl_tile3.wav
pl_duct2.wav	pl_pain4.wav	pl_tile4.wav
pl_duct3.wav	pl_pain5.wav	pl_tile5.wav
pl_duct4.wav	pl_pain6.wav	pl_wade1.wav
pl_fallpain1.wav	pl_pain7.wav	pl_wade2.wav
pl_fallpain2.wav	pl_shell1.wav	pl_wade3.wav
pl_fallpain3.wav	pl_shell2.wav	pl_wade4.wav
pl_grate1.wav	pl_shell3.wav	sprayer.wav
pl_grate2.wav	pl_slosh1.wav	
pl_grate3.wav	pl_slosh2.wav	

B.4.31. roach

rch_die.wav	rch_smash.wav	rch_walk.wav

B.4.32. scientist

_comma.wav	c1a0_sci_dis4a.wav	c1a3_sci_rescued.wav
absolutely.wav	c1a0_sci_dis5a.wav	c1a3_sci_silo1a.wav
absolutelynot.wav	c1a0_sci_dis6a.wav	c1a3_sci_silo2a.wav
administrator.wav	c1a0_sci_dis7a.wav	c1a3_sci_team.wav
afellowsci.wav	c1a0_sci_dis8a.wav	c1a3_sci_thankgod.wav
ahfreeman.wav	c1a0_sci_dis9a.wav	c1a4_sci_blind.wav
alienappeal.wav	c1a0_sci_disa.wav	c1a4_sci_gener.wav
alientrick.wav	c1a0_sci_getaway.wav	c1a4_sci_pwr.wav
allnominal.wav	c1a0_sci_gm.wav	c1a4_sci_pwroff.wav
alright.wav	c1a0_sci_gm1.wav	c1a4_sci_rocket.wav
analysis.wav	c1a0_sci_itsyou.wav	c1a4_sci_tent.wav
announcer.wav	c1a0_sci_lock1a.wav	c1a4_sci_trainend.wav
areyouthink.wav	c1a0_sci_lock2a.wav	c1a4_sci_trust.wav
asexpected.wav	c1a0_sci_lock3a.wav	c2a3_sci_icky.wav
beenaburden.wav	c1a0_sci_lock4a.wav	c2a3_sci_track.wav
beverage.wav	c1a0_sci_lock5a.wav	c2a4_sci_2tau.wav
bloodsample.wav	c1a0_sci_lock6a.wav	c2a4_sci_4tau.wav
c1a0_sci_bigday.wav	c1a0_sci_lock7a.wav	c2a4_sci_alldie.wav
c1a0_sci_catscream.wav	c1a0_sci_lock8a.wav	c2a4_sci_arg2a.wav
c1a0_sci_crit1a.wav	c1a0_sci_mumble.wav	c2a4_sci_arg4a.wav
c1a0_sci_crit2a.wav	c1a0_sci_samp.wav	c2a4_sci_letout.wav
c1a0_sci_crit3a.wav	c1a0_sci_scanrpt.wav	c2a4_sci_scanner.wav
c1a0_sci_ctrl1a.wav	c1a0_sci_stall.wav	c2a4_sci_sugicaloff.wav
c1a0_sci_ctrl2a.wav	c1a0_sci_stayback.wav	c2a4_sci_surgury.wav
c1a0_sci_ctrl3a.wav	c1a1_sci_1scan.wav	c2a5_sci_boobie.wav
c1a0_sci_ctrl4a.wav	c1a1_sci_2scan.wav	c2a5_sci_lebuz.wav
c1a0_sci_dis10a.wav	c1a1_sci_3scan.wav	c3a1_sci_2sat.wav
c1a0_sci_dis11a.wav	c1a1_sci_4scan.wav	c3a1_sci_4sat.wav
c1a0_sci_dis12a.wav	c1a1_sci_5scan.wav	c3a1_sci_6sat.wav
c1a0_sci_dis13a.wav	c1a2_sci_1zomb.wav	c3a1_sci_dome.wav
c1a0_sci_dis14a.wav	c1a2_sci_3zomb.wav	c3a1_sci_done.wav
c1a0_sci_dis15a.wav	c1a2_sci_5zomb.wav	c3a2_sci_1glu.wav
c1a0_sci_dis16a.wav	c1a2_sci_6zomb.wav	c3a2_sci_1surv.wav
c1a0_sci_dis17a.wav	c1a2_sci_dangling.wav	c3a2_sci_2glu.wav
c1a0_sci_dis1a.wav	c1a2_sci_darkroom.wav	c3a2_sci_3glu.wav
c1a0_sci_dis1b.wav	c1a2_sci_elevator.wav	c3a2_sci_3surv.wav
c1a0_sci_dis1c.wav	c1a2_sci_lounge.wav	c3a2_sci_5surv.wav
c1a0_sci_dis1d.wav	c1a2_sci_transm.wav	c3a2_sci_7surv.wav
c1a0_sci_dis2a.wav	c1a3_sci_1man.wav	c3a2_sci_flood.wav
c1a0_sci_dis3a.wav	c1a3_sci_atlast.wav	c3a2_sci_fool.wav

c3a2_sci_forever.wav	greetings.wav	luckwillchange.wav
c3a2_sci_linger.wav	greetings2.wav	madness.wav
c3a2_sci_ljump.wav	headcrab.wav	needsleep.wav
c3a2_sci_notyet.wav	heal1.wav	neverseen.wav
c3a2_sci_portal.wav	heal2.wav	newhevsuit.wav
c3a2_sci_portopen.wav	heal3.wav	newsample.wav
c3a2_sci_position.wav	heal4.wav	nodoubt.wav
c3a2_sci_shower.wav	heal5.wav	nogrant.wav
c3a2_sci_straws.wav	hearsomething.wav	noguess.wav
c3a2_sci_uphere.wav	hello.wav	noidea.wav
cantbeserious.wav	hello2.wav	noo.wav
cantbeworse.wav	hellofreeman.wav	nooo.wav
canttakemore.wav	hellofromlab.wav	noplease.wav
cascade.wav	hellothere.wav	notcertain.wav
catchone.wav	hideglasses.wav	nothostile.wav
chaostheory.wav	holdstill.wav	notsure.wav
checkatten.wav	hopenominal.wav	odorfromyou.wav
completelywrong.wav	hopeyouknow.wav	ofcourse.wav
containfail.wav	howinteresting.wav	ofcoursenot.wav
correcttheory.wav	hungryyet.wav	okgetout.wav
cough.wav	ibelieveso.wav	okihope.wav
delayagain.wav	idontthinkso.wav	organicmatter.wav
didyouhear.wav	ihearsomething.wav	overhere.wav
dinner.wav	illwait.wav	peculiarmarks.wav
dontconcur.wav	illwaithere.wav	peculiarodor.wav
dontgothere.wav	importantspecies.wav	perfectday.wav
dontknow.wav	improbable.wav	perhaps.wav
dontwantdie.wav	imsure.wav	positively.wav
donuteater.wav	inconclusive.wav	protectme.wav
doyousmell.wav	inmesstoo.wav	purereadings.wav
evergetout.wav	ipredictedthis.wav	recalculate.wav
everseen.wav	istay.wav	reconsider.wav
excellentteam.wav	iwounded.wav	repeat.wav
excuse.wav	iwounded2.wav	reportflux.wav
fascinating.wav	iwoundedbad.wav	rescueus.wav
fellowscientist.wav	justasked.wav	ridiculous.wav
fine.wav	lambdalab.wav	right.wav
freeman.wav	leadtheway.wav	rightwayout.wav
freemanalive.wav	leavingme.wav	rumorclean.wav
fusionshunt.wav	letmehelp.wav	rumourclean.wav
getoutalive.wav	letsgo.wav	runtest.wav
getoutofhere.wav	letstrythis.wav	sci_1thou.wav
goodpaper.wav	letyouin.wav	sci_2thou.wav
goodtoseeyou.wav	limitsok.wav	sci_3thou.wav
gottogetout.wav	lowervoice.wav	sci_4thou.wav

sci_5thou.wav	scream08.wav	startle6.wav
sci_aftertest.wav	scream09.wav	startle7.wav
sci_alone.wav	scream1.wav	startle8.wav
sci_bother.wav	scream10.wav	startle9.wav
sci_busy.wav	scream11.wav	statusreport.wav
sci_die1.wav	scream12.wav	stench.wav
sci_die2.wav	scream13.wav	stimulating.wav
sci_die3.wav	scream14.wav	stop1.wav
sci_die4.wav	scream15.wav	stop2.wav
sci_dragoff.wav	scream16.wav	stop3.wav
sci_fear1.wav	scream17.wav	stop4.wav
sci_fear10.wav	scream18.wav	stopasking.wav
sci_fear11.wav	scream19.wav	stopattacking.wav
sci_fear12.wav	scream2.wav	survival.wav
sci_fear13.wav	scream20.wav	thatsodd.wav
sci_fear14.wav	scream21.wav	theoretically.wav
sci_fear15.wav	scream22.wav	thiswillhelp.wav
sci_fear2.wav	scream23.wav	tunedtoday.wav
sci_fear3.wav	scream24.wav	tunnelcalc.wav
sci_fear4.wav	scream25.wav	uselessphd.wav
sci_fear5.wav	scream3.wav	ushouldsee.wav
sci_fear6.wav	scream4.wav	waithere.wav
sci_fear7.wav	scream5.wav	weartie.wav
sci_fear8.wav	scream6.wav	whatissound.wav
sci_fear9.wav	scream7.wav	whatnext.wav
sci_pain1.wav	seeheadcrab.wav	whatyoudoing.wav
sci_pain10.wav	seencup.wav	whoareyou.wav
sci_pain2.wav	shakeunification.wav	whocansay.wav
sci_pain3.wav	shutdownchart.wav	whoresponsible.wav
sci_pain4.wav	shutup.wav	whyaskme.wav
sci_pain5.wav	shutup2.wav	whyleavehere.wav
sci_pain6.wav	simulation.wav	yees.wav
sci_pain7.wav	slowingyou.wav	yes.wav
sci_pain8.wav	smellburn.wav	yes2.wav
sci_pain9.wav	sneeze.wav	yes3.wav
sci_somewhere.wav	softethics.wav	yesihope.wav
scream01.wav	somethingfoul.wav	yesletsgo.wav
scream02.wav	sorryimleaving.wav	yesok.wav
scream03.wav	startle1.wav	youinsane.wav
scream04.wav	startle2.wav	youlookbad.wav
scream05.wav	startle3.wav	youlookbad2.wav
scream06.wav	startle4.wav	youneedmedic.wav
scream07.wav	startle5.wav	youwounded.wav

B.4.33. squeek

sqk_blast1.wav	sqk_die1.wav	sqk_hunt2.wav
sqk_deploy1.wav	sqk_hunt1.wav	sqk_hunt3.wav

B.4.34. tentacle

te_alert1.wav	te_roar1.wav	te_squirm2.wav
te_alert2.wav	te_roar2.wav	te_strike1.wav
te_death2.wav	te_search1.wav	te_strike2.wav
te_flies1.wav	te_search2.wav	te_swing1.wav
te_move1.wav	te_sing1.wav	te_swing2.wav
te_move2.wav	te_sing2.wav	

B.4.35. tride

c0a0_tr_arrive.wav	c0a0_tr_gmorn.wav	c0a0_tr_time.wav
c0a0_tr_dest.wav	c0a0_tr_haz.wav	c0a0_tr_tourn.wav
c0a0_tr_emerg.wav	c0a0_tr_jobs.wav	
c0a0_tr_exit.wav	c0a0_tr_noeat.wav	

B.4.36. turret

tu_active.wav	tu_die2.wav	tu_search.wav
tu_active2.wav	tu_die3.wav	tu_spindown.wav
tu_alert.wav	tu_fire1.wav	tu_spinup.wav
tu_deploy.wav	tu_ping.wav	
tu_die.wav	tu_retract.wav	

B.4.37. vox

_comma.wav	activate.wav	all.wav
_period.wav	activated.wav	alpha.wav
a.wav	activity.wav	am.wav
accelerating.wav	adios.wav	amigo.wav
accelerator.wav	administration.wav	ammunition.wav
accepted.wav	advanced.wav	an.wav
access.wav	after.wav	and.wav
acknowledge.wav	agent.wav	announcement.wav
acknowledged.wav	alarm.wav	anomalous.wav
acquired.wav	alert.wav	antenna.wav
acquisition.wav	alien.wav	any.wav
across.wav	aligned.wav	apprehend.wav

approach.wav	but.wav	d.wav
are.wav	button.wav	dadeda.wav
area.wav	buzwarn.wav	damage.wav
arm.wav	bypass.wav	damaged.wav
armed.wav	c.wav	danger.wav
armor.wav	cable.wav	day.wav
armory.wav	call.wav	deactivated.wav
arrest.wav	called.wav	decompression.wav
ass.wav	canal.wav	decontamination.wav
at.wav	cap.wav	deeoo.wav
atomic.wav	captain.wav	defense.wav
attention.wav	capture.wav	degrees.wav
authorize.wav	ceiling.wav	delta.wav
authorized.wav	celsius.wav	denied.wav
automatic.wav	center.wav	deploy.wav
away.wav	centi.wav	deployed.wav
b.wav	central.wav	destroy.wav
back.wav	chamber.wav	destroyed.wav
backman.wav	charlie.wav	detain.wav
bad.wav	check.wav	detected.wav
bag.wav	checkpoint.wav	detonation.wav
bailey.wav	chemical.wav	device.wav
barracks.wav	cleanup.wav	did.wav
base.wav	clear.wav	die.wav
bay.wav	clearance.wav	dimensional.wav
be.wav	close.wav	dirt.wav
been.wav	code.wav	disengaged.wav
before.wav	coded.wav	dish.wav
beyond.wav	collider.wav	disposal.wav
biohazard.wav	command.wav	distance.wav
biological.wav	communication.wav	distortion.wav
birdwell.wav	complex.wav	do.wav
bizwarn.wav	computer.wav	doctor.wav
black.wav	condition.wav	doop.wav
blast.wav	containment.wav	door.wav
blocked.wav	contamination.wav	down.wav
bloop.wav	control.wav	dual.wav
blue.wav	coolant.wav	duct.wav
bottom.wav	coomer.wav	e.wav
bravo.wav	core.wav	east.wav
breach.wav	correct.wav	echo.wav
breached.wav	corridor.wav	ed.wav
break.wav	crew.wav	effect.wav
bridge.wav	cross.wav	egress.wav
bust.wav	cryogenic.wav	eight.wav

eighteen.wav	first.wav	helicopter.wav
eighty.wav	five.wav	helium.wav
electric.wav	flooding.wav	hello.wav
electromagnetic.wav	floor.wav	help.wav
elevator.wav	fool.wav	here.wav
eleven.wav	for.wav	hide.wav
eliminate.wav	forbidden.wav	high.wav
emergency.wav	force.wav	highest.wav
energy.wav	forms.wav	hit.wav
engage.wav	found.wav	hole.wav
engaged.wav	four.wav	hostile.wav
engine.wav	fourteen.wav	hot.wav
enter.wav	fourth.wav	hotel.wav
entry.wav	fourty.wav	hour.wav
environment.wav	foxtrot.wav	hours.wav
error.wav	freeman.wav	hundred.wav
escape.wav	freezer.wav	hydro.wav
evacuate.wav	from.wav	i.wav
exchange.wav	front.wav	idiot.wav
exit.wav	fuel.wav	illegal.wav
expect.wav	g.wav	immediate.wav
experiment.wav	get.wav	immediately.wav
experimental.wav	go.wav	in.wav
explode.wav	going.wav	inches.wav
explosion.wav	good.wav	india.wav
exposure.wav	goodbye.wav	ing.wav
exterminate.wav	gordon.wav	inoperative.wav
extinguish.wav	got.wav	inside.wav
extinguisher.wav	government.wav	inspection.wav
extreme.wav	granted.wav	inspector.wav
f.wav	great.wav	interchange.wav
facility.wav	green.wav	intruder.wav
fahrenheit.wav	grenade.wav	invallid.wav
failed.wav	guard.wav	invasion.wav
failure.wav	gulf.wav	is.wav
farthest.wav	gun.wav	it.wav
fast.wav	guthrie.wav	johnson.wav
feet.wav	handling.wav	juliet.wav
field.wav	hangar.wav	key.wav
fifteen.wav	has.wav	kill.wav
fifth.wav	have.wav	kilo.wav
fifty.wav	hazard.wav	kit.wav
final.wav	head.wav	lab.wav
fine.wav	health.wav	lambda.wav
fire.wav	heat.wav	laser.wav

last.wav	million.wav	panel.wav
launch.wav	minefield.wav	percent.wav
leak.wav	minimum.wav	perimeter.wav
leave.wav	minutes.wav	permitted.wav
left.wav	mister.wav	personnel.wav
legal.wav	mode.wav	pipe.wav
level.wav	motor.wav	plant.wav
lever.wav	motorpool.wav	platform.wav
lie.wav	move.wav	please.wav
lieutenant.wav	must.wav	point.wav
life.wav	nearest.wav	portal.wav
light.wav	nice.wav	power.wav
lima.wav	nine.wav	presence.wav
liquid.wav	nineteen.wav	press.wav
loading.wav	ninety.wav	primary.wav
locate.wav	no.wav	proceed.wav
located.wav	nominal.wav	processing.wav
location.wav	north.wav	progress.wav
lock.wav	not.wav	proper.wav
locked.wav	november.wav	propulsion.wav
locker.wav	now.wav	prosecute.wav
lockout.wav	number.wav	protective.wav
lower.wav	objective.wav	push.wav
lowest.wav	observation.wav	quantum.wav
magnetic.wav	of.wav	quebec.wav
main.wav	officer.wav	question.wav
maintenance.wav	ok.wav	questioning.wav
malfunction.wav	on.wav	quick.wav
man.wav	one.wav	quit.wav
mass.wav	open.wav	radiation.wav
materials.wav	operating.wav	radioactive.wav
maximum.wav	operations.wav	rads.wav
may.wav	operative.wav	rapid.wav
medical.wav	option.wav	reach.wav
men.wav	order.wav	reached.wav
mercy.wav	organic.wav	reactor.wav
mesa.wav	oscar.wav	red.wav
message.wav	out.wav	relay.wav
meter.wav	outside.wav	released.wav
micro.wav	over.wav	remaining.wav
middle.wav	overload.wav	renegade.wav
mike.wav	override.wav	repair.wav
miles.wav	pacify.wav	report.wav
military.wav	pain.wav	reports.wav
milli.wav	pal.wav	required.wav

research.wav	six.wav	termination.wav
resevoir.wav	sixteen.wav	test.wav
resistance.wav	sixty.wav	that.wav
right.wav	slime.wav	the.wav
rocket.wav	slow.wav	then.wav
roger.wav	soldier.wav	there.wav
romeo.wav	some.wav	third.wav
room.wav	someone.wav	thirteen.wav
round.wav	something.wav	thirty.wav
run.wav	son.wav	this.wav
safe.wav	sorry.wav	those.wav
safety.wav	south.wav	thousand.wav
sargeant.wav	squad.wav	threat.wav
satellite.wav	square.wav	three.wav
save.wav	stairway.wav	through.wav
science.wav	status.wav	time.wav
scream.wav	sterile.wav	to.wav
screen.wav	sterilization.wav	top.wav
search.wav	storage.wav	topside.wav
second.wav	sub.wav	touch.wav
secondary.wav	subsurface.wav	towards.wav
seconds.wav	sudden.wav	track.wav
sector.wav	suit.wav	train.wav
secure.wav	superconducting.wav	transportation.wav
secured.wav	supercooled.wav	truck.wav
security.wav	supply.wav	tunnel.wav
select.wav	surface.wav	turn.wav
selected.wav	surrender.wav	turret.wav
service.wav	surround.wav	twelve.wav
seven.wav	surrounded.wav	twenty.wav
seventeen.wav	switch.wav	two.wav
seventy.wav	system.wav	unauthorized.wav
severe.wav	systems.wav	under.wav
sewage.wav	tactical.wav	uniform.wav
sewer.wav	take.wav	unlocked.wav
shield.wav	talk.wav	until.wav
shipment.wav	tango.wav	up.wav
shock.wav	tank.wav	upper.wav
shoot.wav	target.wav	uranium.wav
shower.wav	team.wav	us.wav
shut.wav	temperature.wav	usa.wav
side.wav	temporal.wav	use.wav
sierra.wav	ten.wav	used.wav
sight.wav	terminal.wav	user.wav
silo.wav	terminated.wav	vacate.wav

valid.wav	warn.wav	xeno.wav
vapor.wav	warning.wav	yankee.wav
vent.wav	waste.wav	yards.wav
ventillation.wav	water.wav	year.wav
victor.wav	we.wav	yellow.wav
violated.wav	weapon.wav	yes.wav
violation.wav	west.wav	you.wav
voltage.wav	whiskey.wav	your.wav
vox_login.wav	white.wav	yourself.wav
walk.wav	wilco.wav	zero.wav
wall.wav	will.wav	zone.wav
want.wav	with.wav	zulu.wav
wanted.wav	without.wav	
warm.wav	woop.wav	

B.4.38. weapons

357_cock1.wav	explode4.wav	pl_gun3.wav
357_reload1.wav	explode5.wav	reload1.wav
357_shot1.wav	g_bounce1.wav	reload2.wav
357_shot2.wav	g_bounce2.wav	reload3.wav
bullet_hit1.wav	g_bounce3.wav	ric1.wav
bullet_hit2.wav	g_bounce4.wav	ric2.wav
cbar_hit1.wav	g_bounce5.wav	ric3.wav
cbar_hit2.wav	gauss2.wav	ric4.wav
cbar_hitbod1.wav	glauncher.wav	ric5.wav
cbar_hitbod2.wav	glauncher2.wav	rocket1.wav
cbar_hitbod3.wav	gren_cock1.wav	rocketfire1.wav
cbar_miss1.wav	grenade_hit1.wav	sbarrel1.wav
dbarrel1.wav	grenade_hit2.wav	scock1.wav
debris1.wav	grenade_hit3.wav	sshell1.wav
debris2.wav	hks1.wav	sshell2.wav
debris3.wav	hks2.wav	sshell3.wav
dryfire1.wav	hks3.wav	xbow_fire1.wav
egon_off1.wav	mine_activate.wav	xbow_fly1.wav
egon_run3.wav	mine_charge.wav	xbow_hit1.wav
egon_windup2.wav	mine_deploy.wav	xbow_hit2.wav
electro4.wav	mortar.wav	xbow_hitbod1.wav
electro5.wav	mortarhit.wav	xbow_hitbod2.wav
electro6.wav	pl_gun1.wav	xbow_reload1.wav
explode3.wav	pl_gun2.wav	

B.4.39. x

nih_die2.wav	x_laugh1.wav	x_recharge2.wav
x_attack1.wav	x_laugh2.wav	x_recharge3.wav
x_attack2.wav	x_pain1.wav	x_shoot1.wav
x_attack3.wav	x_pain2.wav	x_teleattack1.wav
x_ballattack1.wav	x_pain3.wav	
x_die1.wav	x_recharge1.wav	

B.4.40. zombie

claw_miss1.wav	zo_alert20.wav	zo_idle3.wav
claw_miss2.wav	zo_alert30.wav	zo_idle4.wav
claw_strike1.wav	zo_attack1.wav	zo_pain1.wav
claw_strike2.wav	zo_attack2.wav	zo_pain2.wav
claw_strike3.wav	zo_idle1.wav	
zo_alert10.wav	zo_idle2.wav	

C. Changelog

01.01.2010 Erste Fassung

10.03.2012 Links an überarbeitete Seite angepasst.